Automated Tool for Healthcare Data Fusion

Yasmeen AlBarrak

MSc Advanced Computer Science

2013/2014

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) ____________________________
Summary

The goal of this project was to develop a tool that enables the fusion of multiple health-care datasets together and with other open datasets such as census or police datasets. To accomplish this, appropriate methodology that enables multiple datasets to be integrated into a single dataset has been identified with the different possible relationships between the datasets.

Before designing and implementing the project, datasets have been analysed in order to establish the key identifiers in these datasets. This will form the basis for linking the health-care datasets together with datasets from other providers. Moreover, various programming languages and development tools have been searched to select the appropriate technology and libraries to be used in the implementation phase.

From the knowledge obtained in the background chapter, the fusion tool has been designed and implemented. This was followed by a technical and user evaluation to measure the performance and the usability of the developed tool.

Overall, this project aimed to develop a tool for data fusion of health-care datasets with a critical evaluation of the final product.
Acknowledgements

I cannot express enough thanks to my Allah for granting me knowledge, health, and strength to carry on and complete this project.

I would like to thank my supervisor Dr. Roy Ruddle, for his guidance and support throughout my project. I feel truly privileged to have had the opportunity to work with him.

I would also like to thank Dr. Tony Cohn, the project assessor, for his great feedback and guidance in the interim report and progress meeting.

My sincere appreciation to my colleagues and friends, Dr. saja alrayes, Dalal, Salma, and Aisha, for their continued support and encouragements. They were there to cheer me up during the difficult and challenging times in my project.

Finally, I would like to express my sincere heartfelt gratitude and indebtedness to my amazing family, my children, and my husband for providing me with endless love and support. I would have never been able to come and study without their continued support and encouragements.
List of Figures

Figure 1 shows the five phases of the project methodology .......................................................... 4
Figure 2 Example of the CCG and GP hierarchy for Leeds ............................................................ 9
Figure 3 Example of a One-to-One relationship ............................................................................ 10
Figure 4 Example of a One-to-Many relationship ....................................................................... 11
Figure 5 High Level Use Case Diagram ....................................................................................... 18
Figure 6 Screens flow ..................................................................................................................... 19
Figure 7 the activity diagram of the manual fusion scenario ......................................................... 22
Figure 8 the activity diagram of the manual fusion scenario ......................................................... 23
Figure 9 the main window to select datasets .................................................................................. 25
Figure 10 the manual fusion window ............................................................................................ 26
Figure 11 the ‘select columns’ window ......................................................................................... 27
Figure 12 the ‘Finish’ window ....................................................................................................... 27
Figure 13 the project file management in NetBeans ...................................................................... 29
Figure 14 automatic GP and CCG fusion process flowchart ....................................................... 36
Figure 15 Error message for file missing ...................................................................................... 37
Figure 16 Error message for not finding the selected key ............................................................. 37
Figure 17 Confirmation message before closing the tool ............................................................... 37
Figure 18 Error message for not selecting a column in the manual fusion .................................. 38
Figure 19 an example of the profiling tool’s result window ......................................................... 45
Figure 20 methods with highest running time ............................................................................... 46
Figure 21 executing time for fusion method in: (a) automatic fusion for two datasets with common column, (b) automatic fusion for GP and CCG datasets, and (c) manual fusion ................................................................. 48
Figure 22 the size of heap and maximum used heap .................................................................... 49
Figure 23 heap size and maximum used heap using x8 dataset size .......................................... 50
Figure 24 the user evaluation method ........................................................................................... 53
Figure 25 example of the scroll pane used .................................................................................... 55
List of Tables

Table 1 different NHS datasets and the possible key identifiers in each set ........................................ 15
Table 2 use case description for `import datasets and select fusion method` use case ................................ 19
Table 3 Use case description for `Select common column` use case ...................................................... 20
Table 4 use case description for `Select columns to be included in the generated dataset` use case .... 20
Table 5 Use case description for `Save file` use case ........................................................................ 20
Table 6 Use case description for `Exit from the tool` use case ........................................................... 21
Table 7 the tool versions ..................................................................................................................... 30
Table 8 list of methods in “FusionToolUI” Class ................................................................................ 31
Table 9 list of methods in “NextFrame” Class ..................................................................................... 32
Table 10 list of methods in “ManualFrame” Class ............................................................................. 32
Table 11 list of methods in “FinishFrame” Class ............................................................................... 33
Table 12 method in “ReadingFiles” Class ........................................................................................ 33
Table 13 list of methods in “Check_existing_keys” Class ................................................................. 34
Table 14 list of methods in “Automatic_Fusion” Class ................................................................... 34
Table 15 method in “Automatic_Fusion” Class ............................................................................... 35
Table 16 list of tests and their results .................................................................................................. 41
Table 17 list of datasets used in the evaluation .................................................................................. 44
Table 18 shows the performance evaluation for the three cases ..................................................... 45
Table 19 Changes in the error messages .......................................................................................... 54
# Table of Contents

Summary .................................................................................................................. 1

Acknowledgements ............................................................................................... II

List of Figures ......................................................................................................... III

List of Tables ........................................................................................................... IV

**Chapter 1. Introduction** ......................................................................................... 1
  1.1 Overview ........................................................................................................ 1
  1.2 Aim and Objectives ...................................................................................... 2
  1.3 Minimum Requirements and Deliverables .................................................. 2
  1.4 Further Enhancements .................................................................................. 3
  1.5 Project schedule and Progress Report ......................................................... 3
  1.6 Project Methodology .................................................................................... 3
  1.7 Report Outline ............................................................................................. 5

**Chapter 2. Background Research** ...................................................................... 6
  2.1 Background ..................................................................................................... 6
  2.2 Data linkage ................................................................................................... 7
  2.3 Datasets .......................................................................................................... 8
  2.4 Datasets relationships .................................................................................. 10
  2.5 Graphical User Interface .............................................................................. 12
  2.6 Appropriate Technologies ........................................................................... 12
    2.6.1 Programming Language ........................................................................ 12
    2.6.2 Development Tools ............................................................................... 13

**Chapter 3. Solution Design** .............................................................................. 15
  3.1 Data Understanding ...................................................................................... 15
  3.2 Design Consideration ................................................................................... 16
  3.3 Scenario of Usage ......................................................................................... 16
  3.4 User Requirements ....................................................................................... 17
    3.4.1 Use-Case Diagram .................................................................................. 17
    3.4.2 Detail Use-Case Description ................................................................. 19
    3.4.3 Activity Diagram .................................................................................... 21
  3.5 Technologies and Software Architecture .................................................... 24
Chapter 1. Introduction

1.1 Overview

Open data has generated a great deal of improvements in government sectors and organisations around the world. It is considered a tremendous resource that has the potential to revolutionize the delivery of public services, research, and the way governments and organizations work. It may also lead to significant findings that could improve the quality of services and enhance decision-making processes.

Recently, a large amount of data about patients and services, in the course of health and social care services, have been collocated and made available online as open data. It records the daily activities taking place in the different health organisations including audits, surveys and official statistics. This released data has enormous potential in the healthcare sector to improve the outcomes of public services, to promote higher quality and efficient services, and even to drive a significant economic growth [1].

Integrating healthcare data and analysing it can provide valuable knowledge. It is not just about helping to enhance medical knowledge, but also about enabling researchers to identify various clinical trends and healthcare services pathways, and about addressing various research questions. Moreover, fusing healthcare data with other data from different domains, such as census or crimes data, can allow a deeper understanding of the health status and help to define various social economic indicators.

Combining datasets from different sources into a single dataset can be done manually, wherein the data analyst will review the datasets and find match values of specific records, which is consider extremely time consuming. However, using a software that automatically fuses datasets from multiple sources will reduce the time needed for the manual review and produce results in much more efficient and faster way. The next chapter shall describe and discuss the possible approaches.

The national provider of information, data and IT systems for health and social care (HSCIC) has published data collected from across the health and social care systems in the UK. This open data from around 680 different data sources is available from the HSCIC website. This was the main driver to build a tool that could merge healthcare datasets in an automatic way.
1.2 Aim and Objectives

The aim of this project is to develop a tool that enables the fusion between healthcare datasets (HSCIC datasets) and other open datasets, such as from census, weather and geographic sources, into a single dataset for the purposes of data mining and analysis. The link between these different datasets will be mainly geographic. In order to achieve this fusion, an appropriate methodology enabling multiple datasets with varying data models to be integrated into a single dataset will be used. In order to accomplish this aim, the following objectives must be achieved:

1. Identify appropriate open datasets that will be linked to the healthcare dataset.

2. Identify appropriate methods to link the datasets, covering multiple datasets from a single provider (e.g., HSCIC) and from several providers (e.g., HSCIC and census).

3. Encapsulate within the tool information of where different datasets are, to promote ease of access.

4. Produce examples of integrated data output in an appropriate format.

5. Perform technical and user evaluations of the tool.

1.3 Minimum Requirements and Deliverables

The following are the minimum requirements set for the project.

1. Identify fusion or link techniques that will be suitable for linking multiple datasets e.g. deterministic linkage methodology.

2. Develop a tool that can fuse two user-provided datasets (e.g., HSCIC datasets) in a CSV format and produce the resultant single dataset in a flat file text format, given dataset field mappings either contained within the tool or provided by the user.

3. Develop a graphic user interface (GUI) that allows a user to import two datasets in a CSV format and generate the output.

4. Allow the user to choose the fusion method, namely automatic (where the tool will find the way to fuse the datasets) or manual (where the user can specify the common column in both datasets).
1.4 Further Enhancements
The possible extensions are:

1. Enhance the tool to fuse a healthcare dataset with Police or Census datasets.

2. Enhance the algorithm to fuse two datasets with implicit links between them.

1.5 Project schedule and Progress Report
To ensure a good understanding of the project requirements and the successful progression of the data analysis, a weekly meeting has been scheduled with the supervisor from March 2014. Prior to this, a meeting with Dr. Chris Smith, senior Research Fellow and data scientist at the Leeds Institute of Health Sciences, who posted this project as an MSc project, was conducted to give an overview of the project.

As agreed with the supervisor, the main focus in the initial stage of the project was the data analysis of the healthcare datasets. This includes reading about and understanding the hierarchy of the healthcare systems in the UK, finding identifier elements in each dataset, such as postcode or organization code, and find a link between these elements. For instance, to define the relationship between the General practices and the Clinical Commissioning Group (CCG).

A project plan has been set up to help to organise and manage the time needed to complete each phase of the project. However, the full implementation took more time than was originally allocated in the project plan. This led to some delay at the start of the evaluation phase of the first iteration, and to the impossibility to complete the second iteration of the project. A copy of the project plan can be seen in Appendix C. A presentation was also provided at the progress meeting in July, which is attached in appendix D.

1.6 Project Methodology
The project aimed to develop a tool that could be used for fusing two healthcare datasets. Various software development methodologies have been reviewed in order to build the tool. The System Development Life Cycle (SDLC), a conceptual model used in project management, was chosen. The rational for choosing this method is that it seems to be the best fit for this project compared to other methods. SDLC is composed of several well-defined stages involved in an information system development, from planning the design to achieving full completion of the project.
Based on the SDLC method, the project process can be broken down into five phases:

1) Requirement phase: to understand the healthcare datasets. This includes background reading about the health system in the UK, as well as engaging an expert to provide a better understanding of the health datasets and of the different entities within the data.

2) Analysis phase: to analyse the data and find the key identifiers in each dataset, which will then be used to link these datasets.

3) Design phase: to determine appropriate programming languages and tools for the data fusion. This phase also includes designing the system architecture, demonstrating different use cases, and building a user interface for the tool.

4) Implementation phase: where the actual program code will be written.

5) The final phase in this iteration is the evaluation phase, which involves technical and user evaluation of the tool to ensure that the tool functions appropriately.

Figure 1 shows the five phases of the project methodology
1.7 Report Outline

In chapter 2, a general background to the problem and the suitable methods for the implementation of the tool is presented. It will provide a review for the definition of data linkage and its different techniques, a summary of the datasets used in this project, an exploration of the different possible relationships between the datasets, and the appropriate technologies to be used in the implementation of the project.

Chapter 3 describes the design phase of this project. It starts with analysing the healthcare datasets used in this project as well as the design consideration related to the datasets. Then, a different scenario of usage and the user requirements, including the use-cases diagrams and activity diagrams, are presented. Further into the chapter, the technologies and the software architecture will be discussed. Finally, the user interface design characteristics that have been considered and screenshots of the interface are described.

Chapter 4 deals with the actual implementation of the project for the two subsystems, namely the graphical user interface and the back-end. This was followed by a test section explaining the test process that has been carried out during the implementation.

In chapter 5, the evaluation phase of the project is explained. The evaluation is divided into two main sections, technical evaluation and user acceptance evaluation.

Finally, chapter 6 serves as a conclusion for the project, evoking the remaining challenges and outlining the limitations of this project.
Chapter 2. Background Research

This section aims to provide a general background to the problem, to describe techniques that will be used in this project and to summarize details of the datasets that the project will be dealing with.

2.1 Background

The past several years have seen an exponential increase in the number of published health and social care data. A vast amount of data about patients and services in the health domain have been collected and recorded routinely through the healthcare systems [2]. This aggregated data has enormous potential in terms of health-related research, in tackling healthcare questions, in improving the quality of the services, in strategic service planning and improving clinical decision-making and service delivery [3, 4].

The national provider of information, data and IT systems for health and social care (HSCIC) has published data collected from across the health and social care systems in the UK. It can therefore be linked and analysed in ways that can improve the outcomes of care and public services [1]. This open data, which comes from around 680 different datasets, is available from the HSCIC website and can be accessed by patients, researchers, universities and other organisations.

The generated dataset contains official statistics, surveys, audits, and other reports and statistics that are generated quarterly or yearly [1]. They are structured into lists of individual data items, each with a clear field name, definition, and set of admissible values, codes and classifications [1]. From this, valuable information can be derived, which can then be used to monitor and evaluate healthcare services.

Establishing links between two of the healthcare datasets, or with other types of sets such as census, crimes, or geographic data, provides the ability to analyse the different care services pathways in the UK, and to address complex research questions, involving the entire context around the data rather than just the data itself, that would consequently not be properly addressed with a single dataset. It helps to identify trends and variations in patient outcomes, social economic indicators, or some measurements such as the phenomenon of deprivation in some regions. In other words, it enables a much deeper understanding of the healthcare pathway [1].

One linking method is to manually review both datasets and compare them, which is extremely time consuming. The data analyst has to compare the different records in the datasets and try to identify fields to link together, as well as find matching records. However, using a software that can automatically match
the records between two datasets and fuse them into a single dataset can make results more efficient and faster. Using computers for exact matching will reduce or eliminate manual review, as well as produce timely results of better quality, consistency and reproducibility [5].

The basic linking process of two datasets involves the use of identifiers that are available in both sets of records [6]. These identifiers can include name, address or a specific ID number. More efficient linkage is usually achieved by using all available means of identification [6]. In healthcare datasets, the code number representing the different healthcare systems is mainly used as the key identifier. Examples of these codes are General Practice (GP), Clinical Commissioning Group (CCG), Network or Trust codes. On the other hand, linking healthcare dataset with other data types such as census or crime will be mainly geographic, using postcodes, Lower layer super output area (LSOA) or ONS (office for national statistics) codes.

2.2 Data linkage

Data linkage is a general term that covers a very wide domain. Several terms have appeared that have the same meaning, such as merging, combining, fusion and integrating [7]. These refer to linking or merging data from multiple sources to obtain comprehensive information and to conduct more powerful data analysis and data mining. Data linkage is an important data pre-processing step in data mining projects, and takes about 50% to 70% of the time and effort required in any data mining process [8]. The aim of such linkage is to match all records from the different datasets related to the same entity, such as a given individual or organization [8].

Data linkage allows re-use of existing datasets from multiple sources for new studies, and enriches the data used for pattern detection in data mining [8]. For example, in the health sector, linking data may provide information that can help to improve health services and policies.

Linking two datasets for research purposes can be done using distinct linkage techniques. These different techniques can be broadly classified into deterministic and probabilistic approaches. Deterministic linkage involves a full agreement or exact one-to-one matching of a unique identifier or a set of common identifiers [9]. It requires high quality formal identifiers in order to achieve good linkage results [8]. This method of linkage ensures more certainty in the match between two datasets, since only a complete match on a set of identifiers is accepted [9].

Probabilistic linkage combines two records from two datasets based on various criteria. It calculates the statistical probability of a set of common identifiers to decide whether two records should be combined
and whether they refer to the same entity [9]. Thus, probabilistic linkage “maximises linkage theoretically” and increases the uncertainly of some links[9].

As the healthcare datasets that will be used in this project have clearly identified, for each record, a code from the healthcare system such as a General Practice (GP) code that can be used as an identifier, deterministic linkage seems to be the best fit for the first iteration of the project, which aimed to link two healthcare datasets. For the second iteration, which aimed to link health-related datasets with other non-health-related datasets, the deterministic approach can be used for the linkage based on, for example, the postcode as an identifier.

2.3 Datasets

A number of health care datasets that have been published by the HSCIC under the terms of the open government licence will be used as test data in this project. This data is being made available as comma-separated values (CSV) files. This released data includes:

1. General practice prescribing data, which covers all the medicines, appliances and dressings that are prescribed by each practice in England.
2. Number of patients registered at a GP practice, provided by GP code, CCG code or by Lower Layer Super Output area (LSOA) code.
3. Data from National clinical audits, such as National Bowel cancer and lung cancer audits, National Diabetes Inpatient Audit (NaDIA).

This data is structured to include key identifiers to represent each row in the datasets. Identifiers have to be unique in order to identify each row. For example, Prescribing datasets and Number of patients registered at a GP practices are structured to include GP codes or CCG codes as key identifiers, while the data from different national cancer audits is structured to include Network or trust code as key identifiers. Details about the datasets and the identifiers in each dataset are described in the next chapter.

In the UK, health and social care are provided by multiple systems, each with a unique code. General Practices (GPs) is a primary care unit that represent the first point of contact for most patients and provide a variety of health services within the community. There are currently around 8,230 practices in England (The Information Centre, 2009). Data extracted from a GP could give an indication of the health status in the community wherein the GP is located. GP practices are identified only by their unique national code.
Clinical Commissioning Groups (CCGs) are National Health Service (NHS) organisations that organise the delivery of a group of GPs in their geographical area. Thus, each GP has to work under a particular CCG which has a unique CCG code. For example, NHS Leeds West CCG includes all the GPs in the west area of Leeds; as such, any GP with a postcode located in the boundary of the west Leeds region will be under this CCG. This hierarchy in the healthcare system seems to be as a key to a successful linkage between the datasets in the health domain. Figure 2 shows an example of the CCG and GP hierarchy for Leeds and gives some examples of codes used within this structure.

![CCG and GP hierarchy for Leeds](image)

**Figure 2 Example of the CCG and GP hierarchy for Leeds.**

Other datasets from different domains can be merged with healthcare data, which will be tackled in the second iteration of the project, to infer other social or economic indicators. Examples of this possible data are census and crime datasets. The linkage of this data will be mainly geographic, linking census datasets based on postcodes which can also be found in some of the healthcare datasets. The police also provides datasets about street-level crime, broken down by Lower Layer Super Output Area (LOSA) codes which can be used to link to some of the health care datasets.

Linking this vast amount of data with such different codes and thousands of records manually would be extremely time- and effort-consuming. This includes understanding the context for both datasets, identifying the key to link the datasets together, and matching each possible pair of identifiers within the records. The task can become even more complicated if the key identifiers of both datasets are different, such as GP codes in one and CCG codes in the other.
2.4 Datasets relationships

Relationships between heterogeneous data sources exist when these sources have common data, either on a physical or on a logical level [10]. For example, two datasets that contain the same fields, such as GP codes, can be linked through this common field containing matching values. On the other hand, another dataset could contain a field such as CCG code values, which are logically related to the GP code field in the first dataset. The relationship between CCG and GP codes will be described later in this chapter.

Therefore, identifying the relationship between datasets is crucial knowledge and an important step before linking the data. It helps to understand how to process this data in order to obtain the desired result [10]. Relationships between datasets can be one of four categories:

- **One-to-One relationship**

  In this type of relationship, a single value of a selected column in one dataset is related to a single value in another column from the second dataset. This implies that each value from the selected columns, used to identify the relationship between the two datasets, exists just once in each dataset [10]. For example, a dataset containing the number of patients registered in each GP and a dataset of a GP practice index of multiple deprivations are linked by the common value of the GP code. Figure 3 illustrates a One-to-One relationship.
**One-to-Many or Many-to-One**

This kind of relationship between datasets implies that one of the datasets contains a single value in a specific column that is related to multiple values in a specific column in another dataset. For instance, the relationship between GPs codes and CCGs codes in the healthcare datasets is considered one-to-many, as each CCG supervises the work of multiple GPs in a given geographic area. The relationship can be One-to-Many or Many-to-One based on “the order in which datasets are processed” [10]. Figure 4 illustrates an example of One-to-Many relationship.

![Figure 4 Example of a One-to-Many relationship](image)

**Many-to-Many**

A Many-to-Many relationship implies that multiple values of a selected column in a dataset can be related to multiple values in another dataset. For example, the relationship between GP datasets and prescribing datasets, each containing several drugs names, is considered a Many-to-Many relationship. For a particular period of time, a GP can prescribe several drugs while on the other hand, a drug can be prescribed by multiple GPs.
2.5 Graphical User Interface

A graphical user interface (GUI) is a “human-computer interface” that allows users to interact with electronic devices using windows, icons and menus [11]. It is the most important part of a system from the user perspective and it is where the user can act and see the responding action from the system. Thus, a well-designed interface can affect the user’s choice of particular software [12].

There are various interface-design methodologies that can help developers make an efficient and effective user interface. These methodologies include iterative sketch processing, prototyping and collecting user feedback. However, these methodologies cannot be useful unless developer know the users’ needs. This includes understanding the users’ goals, skills, and the main functions that they want the software to perform.

2.6 Appropriate Technologies

This section describes the appropriate technologies that can be used in the implementation of this project.

2.6.1 Programming Language

An important task in creating a software solution is choosing the appropriate programming language that will be used to represent it. There are multiple high-level programming languages that are widely used and can be considered in this project. However, a suitable language for the implementation of the fusion tool should be capable of the following:

1. Create a Graphical User Interface (GUI).
2. Read data from a CSV file.
3. Generate the output in Flat file text format.

Many popular languages can satisfy these conditions, but the two languages that were more seriously considered which are Java and C++. They are general purpose languages that work on many platforms. Java is used in a wide number of computing platforms. It provides programmers with a certain ease of use to develop generic applications, with many dynamic libraries devoted to different usable functions, and many online tutorials on how to use them. For example, javax.swing is a library specifically devoted to graphical user interfaces.
C++ was designed for system and application programming. It provides a standards library which includes generic containers and algorithms. However, C++ does not include a built-in standard GUI library. Therefore, a cross-platform application and UI framework can be used to solve this lack of UI library. For example Qt, which is an open source project for multi-platform applications and UI framework for developers using C++.

### 2.6.2 Development Tools

There is a variety of development tools with different degrees of complexity that can be used to create, debug, and maintain an application or a tool. A developer or a team can decide which development tool to use based on their personal preference, their experience of a specific tool, or the size of the problem they want to solve.

The most basic and simplest tool to write a program is a text editor, which is often provided with operating systems such as Microsoft’s Notepad. Text editors can be used to read, create, and modify a plain text file using simple characters, and they have a limited functionality, not allowing developers to write many lines of programming. Some other text editors such as vi or Emacs for Unix operating systems can offer broader and more complex functionality, such as syntax matching for popular programming languages. Although a text editor can be used to program a small application, a programmable editor is a preferable option for a more substantial program.

Integrated Development Environments (IDE) is a development tool that provides comprehensive features to programmers in order to run their program. It consists of single program with all the simple and complicated functionalities needed to develop an application. IDEs provide features for writing source code, modifying, compiling, deploying and debugging a program. NetBeans, Eclipse, Microsoft’s Visual studio are widely-used examples of IDEs tool.

NetBeans IDE is free, open-source, and can be used easily to develop desktop, mobile, and web applications with multiple languages like Java, HTML, PHP, and C/C++ [13]. NetBeans is supported across different platforms and can be run on all operating systems supporting Java such as Windows, OS X, and Linux. It offers a large amount of features to build desktop applications, such as user interface management, storage management, and NetBeans Visual library [13]. In addition, it helps developers to organise their folders and files and allows them to open multiple files from different projects. NetBeans will be used in the implementation of this project.
Eclipse is another popular example of IDE. It is an open-source tool that allows developers to test code and structure their folders, and it additionally provides an intelligent text editor [14]. It supports multiple programming languages such as Java and Python, and has an embedded collection of plugins such as Google’s Android to develop Android applications [14]. Another IDE preferred by some developers is Microsoft’s Visual Studio, which allows the concept of visual programming. User interfaces and their objects are created visually rather than by writing lines of code.
Chapter 3. Solution Design

This chapter explains the design phase of the project. This includes the programme architecture with detailed descriptions of different use cases and process diagrams considered during the user-interface design process.

3.1 Data Understanding

To build the prototype, we developed a fusion algorithm based on deterministic linkage between the datasets. This technique requires a high-quality formal identifier to achieve good linkage results [8]. For this reason, it was necessary to analyse the healthcare datasets first to find the key-identifiers to be used in the fusion method for the first iteration of the project. Thus, we included a list of all possible healthcare datasets with a description of each set and the key identifier of each set (see Table 1).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Possible Identified key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of patients registered at GP Practice By GP</td>
<td>GP Practice Code</td>
</tr>
<tr>
<td></td>
<td>Postcode</td>
</tr>
<tr>
<td></td>
<td>CCG code</td>
</tr>
<tr>
<td>Number of patients registered at GP Practice By CCG</td>
<td>CCG code</td>
</tr>
<tr>
<td>Number of patients registered at GP Practice By LSOA</td>
<td>LSOA</td>
</tr>
<tr>
<td>GP practice Prescribing Presentation level</td>
<td>GP Practice Code</td>
</tr>
<tr>
<td>CCG prescribing data</td>
<td>CCG code</td>
</tr>
<tr>
<td>GP practice Prescribing chemical Level</td>
<td>GP Practice Code</td>
</tr>
<tr>
<td>NHS dental Statistics for England</td>
<td>CCG code</td>
</tr>
<tr>
<td>Index of multiple deprivation (IMD)</td>
<td>GP Practice Code</td>
</tr>
<tr>
<td></td>
<td>PCT code</td>
</tr>
<tr>
<td>National Bowel Cancer audit</td>
<td>Cancer network</td>
</tr>
<tr>
<td>National Diabetes Inpatient audit (NaDIA)</td>
<td>Provider code</td>
</tr>
<tr>
<td>National Lung Cancer audit</td>
<td>Trust code</td>
</tr>
<tr>
<td>National Head Neck Cancer audit</td>
<td>Organisation code</td>
</tr>
</tbody>
</table>

Table 1 different NHS datasets and the possible key identifiers in each set

After reviewing the datasets, it was very clear that the key identifier is the unique number of the NHS organisation that provides those datasets. These organisations include GPs, CCGs, Hospitals, NHS trusts and cancer centres. Some of the datasets could have more than one column with unique values which can be used as a key identifier for each record. For example, datasets of the number of patients registered at GP practice include GP code, post code, and CCG code. Each of these datasets contains unique values that can be used to merge the data with other sets that contain one of these columns.
Although datasets are provided by a single provider, the HSCIC website, these data are not consistent. There are no standards for the field names on these datasets, especially the key identifier columns. Some datasets used just “Code” to refer to the organisation that provides these data while some datasets use the organization name alone or followed by the word “code”. For example, GP code can be referred to as “Practice”, “Practice_code”, or “Practice code”.

All of these observations were considered before designing the prototype. This helped make a robust design with the main functionalities to achieve the minimum requirements as described in this section.

3.2 Design Consideration

The HSCIC datasets used for this project are available in CSV format. These datasets are structured data with defined columns and one or more identifier column which, after analysing the datasets, could be GP code, CCG code, Post code, or ONS code. Merging the HSCIC datasets mainly is based on those key identifiers and finding a linkage between them. Thus, there are some standards for the datasets so that the developed tool can accept and perform the fusion:

- Must be in CSV format
- Must include at least one of the identifiers
- Must include a link between the identifiers in the datasets to be merged. For example, GP code could link with CCG code, as each GP worked under a particular CCG.

3.3 Scenario of Usage

For the design, four usage scenarios were considered. The first and second scenarios are when the user selects the automatic-fusion method, the tool will search for the predefined unique key, which is GP code, CCG code, or Trust code, to perform the fusion. The third scenario is when the user chooses to do the manual-fusion method, which allows the user to select the common column to perform the fusion across all datasets. The last scenario is when the selected two datasets do not match and the tool could not create a mapping between them.

In the automatic-fusion scenario, there are two use cases. The first case is the automatic fusion of two datasets that contain the same unique entity identifier (key column). For example, both datasets contain the CCG code. In this case, the problem of merging at the entity level becomes trivial. It is a simple
matching between the values of these two keys taking advantage of the standard formats of these key fields.

The second case is the automatic fusion of two datasets without having a common unique key between them, but while there is still a relationship between those keys. For example, in the healthcare datasets, although GP code and CCG code are different unique codes that represent different organisations in the United Kingdom health system, there is a relationship between these two keys which enable the mapping between them. Each GP has to work under a particular CCG which has a unique CCG code. Thus, having this relationship implicit in the tool as a list of all the CCG codes and its list of GPs will make the fusion between these datasets possible.

The third scenario that the tool has to handle is the manual fusion of two datasets. Users can choose a particular common column from the datasets to use for a fusion across the datasets. This can be done by providing users with a list of columns in the first and second dataset and allowing the user to choose the column. Then, the tool has to decide if there is match between the selected columns and whether to proceed with the fusion method or show the user-error message.

In the final scenario, when the selected two datasets do not match and the tool could not create a mapping between them, an error message should pop up for the user that the fusion can’t be completed.

3.4 User Requirements

The user requirements for a system is the structured representation of functional and non-functional requirements that should be explained in a way that the system users can understand [15]. It describes without detailed technical knowledge the main functions or activities that the user can perform using the developed software. The user requirements should be written in simple language and using simple tables and diagrams [15]. For this project, use-case diagrams, tables for the detailed use-cases, and activity diagrams are used to describe the user requirements.

3.4.1 Use-Case Diagram

A use case diagram is a way to model the interaction between a system and its intended users (actors) without providing the details [11]. It helps the developer to a better understanding of the system’s intended behaviour. Based on the usage scenarios that were considered for the design, set of main activities that a user can perform using the tool were set out in a high level use case diagram (See Figure 5).
Based on these activities that a user can perform using the tool, it was decided to have four different screens for the user interface. In the first screen, the user imports two datasets, selects the key identifier for each dataset, and selects the fusion method. If the user has selected the automatic fusion method, then a second window appears to allow the user select columns from both dataset to include in the integrated dataset. On the other hand, if the user has selected the manual fusion methods then another screen appears which allow the user to select common column from each dataset to use for fusing the datasets. Then, the
user again can select columns from both dataset to include in the integrated dataset. After generating the dataset, the user can save the generated dataset in a CSV file and a fourth screen appears to enable the user to close the tool or to use the tool again. Figure 6 illustrate the screens flow.

![Figure 6 Screens flow](image)

### 3.4.2 Detail Use-Case Description

Detailed use cases were produced for each use case in the above diagram.

a) **Use Case Name: Import datasets and select fusion method**

<table>
<thead>
<tr>
<th>Description:</th>
<th>User imports two CSV files in the first window of the user interface in order to merge them into a single dataset.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precondition:</td>
<td>None.</td>
</tr>
</tbody>
</table>
| **Basic Flow of Events:** | 1. The system displays the first window where the user imports two datasets and chooses a fusion method.  
2. The user selects “Browse” for the first dataset.  
3. The system displays an “Open File” window to select the directory and the dataset.  
4. The user selects “Open” and the file path will appear in the text field.  
5. The user selects the key from a key list.  
6. The user repeats steps 2, 3, 4, and 5 for the second dataset.  
7. The user selects “Next”. |
| Post condition: | None.                                                                                                         |

Table 2 use case description for `import datasets and select fusion method` use case
b) **Use Case Name: Select common column**

<table>
<thead>
<tr>
<th>Description</th>
<th>User selects the common column in both datasets to use in matching across the datasets.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precondition:</strong></td>
<td>The user has to select the manual-fusion method in the first window in order to choose the common column.</td>
</tr>
</tbody>
</table>
| **Basic Flow of Events:** | 1. The system displays the manual-fusion window containing two lists of columns from the first and second datasets.  
2. The user selects the common column from the first and second datasets.  
3. The user selects “Next” to proceed to the next window. |
| **Post condition:** | None. |

Table 3 Use case description for `Select common column` use case

---

c) **Use Case Name: Select columns to be included in the generated dataset**

<table>
<thead>
<tr>
<th>Description</th>
<th>User selects the columns from both datasets to include in the generated dataset.</th>
</tr>
</thead>
</table>
| **Precondition:** | The user has to choose the common column for manual fusion, or  
The user has to select the datasets and automatic-fusion method from the first window. |
| **Basic Flow of Events:** | 1. The system displays the automatic-fusion window containing two lists of columns from the first and second datasets.  
2. The user selects columns from the first and second datasets that he/she wants to include in the new generated dataset or by default selects all columns.  
3. The user selects “Next” to proceed to the next window. |
| **Post condition:** | None. |

Table 4 use case description for `Select columns to be included in the generated dataset` use case

---

d) **Use Case Name: Save file**

<table>
<thead>
<tr>
<th>Description</th>
<th>User saves the generated dataset as a CSV file in a selected directory.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precondition:</strong></td>
<td>The user selects the columns to include in the generated dataset and then selects “Next”.</td>
</tr>
</tbody>
</table>
| **Basic Flow of Events:** | 1. The system displays a “Save As” window to let the user select a directory to save the new file.  
2. The user selects the directory in the computer.  
3. The user selects “Save”. |
| **Post condition:** | None. |

Table 5 Use case description for `Save file` use case
c) **Use Case Name: Exit from the tool**

<table>
<thead>
<tr>
<th>Description:</th>
<th>The user can exit from the tool any time.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precondition:</strong></td>
<td>None.</td>
</tr>
</tbody>
</table>
| **Basic Flow of events:** | 1. The system displays a “Cancel” option in the bottom right corner in each window in the user interface.  
2. The user can select “Cancel” and the system will generate a dialog box to confirm if the user wants to close the tool or not.  
3. If the user select ‘Yes’ in the dialog box, the tool will automatically close. |
| **Post condition:** | None. |

Table 6 Use case description for `Exit from the tool ` use case

### 3.4.3 Activity Diagram

Activity Diagram visually illustrates the workflow of activities that user goes through as a part of a larger procedure [16]. It can be used to model a specific use case in more detailed level [16]. For this project, activity diagrams have been used to model two main scenarios namely the manual fusion and automatic fusion method. Each scenario contains use cases that in combination form a complete scenario. Figure 7 and figure 8 show activity Diagrams for Manual and automatic fusion scenarios.
Figure 7 the activity diagram of the manual fusion scenario
Figure 8 the activity diagram of the manual fusion scenario
3.5 Technologies and Software Architecture

A number of Programming languages and development tools were consider for the implementation of this project that can satisfy the criteria derived from the minimum requirements of the project (as discussed in 2.5.1). The language chosen for the project is Java because it has large standard libraries with rich functionalities comparing to C++ standard library. In addition, Java development tools provides robust GUI framework with wide range of useful tutorials and source documents that are available on-line.

The system is separated into two subsystems: Graphical User Interface which represents the front-end of the application and fusion which represents the back-end. The GUI provides the user with the functionality to import the datasets in a CSV format and to select the fusion method. This can be automatic fusion where the tool will find the way to fuse the datasets or manual fusion where the user will specify the common column in both datasets. The GUI will pass on user input to the fusion subsystem which deals with finding the linkage and create the new file with data merged from both datasets. The fusion subsystem then returns the output file to the user in a single CSV format.

3.6 Graphical User Interface (GUI)

In this project, users’ main concern is to create a tool that can perform fusions between datasets from different resources. These tools have to be direct, straightforward, and effective at the same time. Therefore, some characteristics were considered in designing the user interface of the tool in order to achieve the simplicity and effectivity. These characteristics are listed below.

**Clear and simple:** The user interface has to be clear and simple. Clarity is the main purpose of design that allows users to interact with the system, understand how the system works, and navigate easily without confusing users [17]. Thus, the tool user interface is designed to let the user perform the data fusion in simple steps. Each step has its own window with a simple design. For example, the first step in fusing two datasets is to import the datasets and select a fusion method, which will be presented in a single window. Then the user has to press a “Next” button to move to the following step. Screen layout will be presented later in this section.

**Familiar:** The user interface has to be familiar and intuitive. This means designing an interface that appears similar to something the users have encountered before [17]. This was accounted for in the tool windows by using familiar names and positions for the buttons in the window. For example, control buttons are located at the bottom-right corner of the screen and use familiar names such as “Next”, “Back”, “Cancel”, and “Browse”.
Responsive and immediate feedback: The user interface has to be fast and give the user immediate results or feedback for some actions. For example, a dialog box appears to ask the user to select a fusion method if the user presses the “Next” button without selecting one.

Consistent: For the tool user interface design, consistency was taken into account. Windows were designed to include consistent terms, including the error messages, to make it easy for the user to understand the error and try to fix it. For instance, the term “key” is used to refer to the identifier column that the system uses to perform the fusion in the first window. Therefore, the term “key” is also used in the error message to inform the user that one of the key columns was not found in the datasets and therefore the fusion process cannot be completed.

3.6.1 User Interface design

Taking in the consideration the above characteristics of the user interface, the tool user interface prototypes were built as follows.

The main window in the software is the first step of data fusion. A user has to select two datasets and a key which represents the NHS organisation that provided the data. This will help the tool determine the predefined key field in each dataset and decide if it is able to merge these datasets. In addition, the user has to select one of the fusion methods which will be, by default, an automatic-fusion method. Figure 9 shows the main window used to select two datasets.

![Automated Fusion Tool](image)

Figure 9 the main window to select datasets
The second window appears after the user has selected a manual-fusion method. It allows the user to choose the common column in the datasets. The column names of each dataset will be listed using a list of radio buttons where the user can select one column from each dataset then press “Ok” to move to the next step. Figure 10 shows the manual-fusion window.

![Figure 10 the manual fusion window.](image)

The third window appears after the user presses the “Next” button from the manual-fusion window or if the user has selected the automatic-fusion method from the first window. This window enables the user to select which fields will be generated in the single dataset. These fields are listed using checkboxes to allow the user to select more than one column from both datasets. In the default situation, all fields from both datasets are selected for inclusion in the generated dataset, and the user can just remove the check from the “Select all” box at the bottom of the column list. Figure 11 shows the “Select columns” window.
Figure 11 the ‘select columns’ window.

The last window is to confirm that the dataset has been generated and saved in the selected directory. It also contains two options for the user: either to finish and close the software, by pressing the “Finish: button, or to use the software again by opening the main window using the “New” button. Figure 12 shows the “Finish window”.

Figure 12 the ‘Finish’ window.
3.7 Back-end Design

The user interface is designed to obtain data and process user actions by passing these to the back-end which, in turn, will process the data and return the results to the user. The back-end contains all the classes, methods, and objects necessary to complete the required job. For example, the back-end should contain methods; to read CSV files and load the data in a list, arrays, or another data types, to check the selected keys exist in the datasets, to fuse the datasets automatically or manually, and to generate the datasets in a CSV files. The list of all classes and methods in the back-end subsystem are explained in the implementation chapter.

3.8 Chapter Summary

The chapter discusses the design phase for the development of the Automated Fusion Tool. This includes the data understanding as a preparation stage before designing the solution. Then, a list of scenarios of usages and the user requirements are explained in order to guide the developer in this phase. In addition, this chapter set outs the graphical user interface and some of its characteristics to consider while developing the user interface prototypes.
Chapter 4. Implementation

This chapter explains the implementation phase for the project. In the implementation phase, the actual software was built and installed. Taking into account the requirements and the design considerations, the automated fusion tool was developed. A description of the classes and methods created to build the tool is provided in the earlier part of this section. A full documentation about the classes has been generated from the source code, in an HTML format, by using the Javadoc tool, attached in Appendix H.

Following the implementation of the project, a set of software tests were performed. The testing is an investigation process to improve the quality of the developed software and to provide evidence that it is working correctly. A description of the testing stage appears later in this chapter.

4.1 Development Tool

The software was implemented in NetBeans by using Java programming language as discussed in section 3.2. Use of NetBeans has helped to build the GUI of the software due to its built-in features and visual libraries. The file management is one of the NetBeans features that was helpful in the development phase of this project as it helps the developer to organise the files and open multiple files from different projects. Figure 13 shows the project file management in NetBeans.

![Figure 13: the project file management in NetBeans.](image-url)
4.2 Version control

The development of the software was divided into a number of versions. Each version was tested before development of the next stage could continue. The versions are described in Table 7.

<table>
<thead>
<tr>
<th>Version</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Prototype of the main window</td>
<td>This version includes the main interface and its functionalities such as reading two CSV files and selecting a key and the fusion method.</td>
</tr>
<tr>
<td>0.2</td>
<td>Prototype of the main window and the selecting columns window</td>
<td>This version includes the next window for the automatic fusion method where the user selects columns to add to the dataset that will be generated. This window appears by clicking the Next button in the previous window.</td>
</tr>
<tr>
<td>0.3</td>
<td>Previous prototypes and the automatic fusion method</td>
<td>In this version, a method for integrating two datasets has been created to be called from the class of the second window.</td>
</tr>
<tr>
<td>0.4</td>
<td>Adding the window for the manual fusion method</td>
<td>This version includes the manual fusion window and its functionalities such as listing the column headers of both datasets and selecting common columns between datasets. This window appears by selecting Manual as the fusion method and clicking the Next button from the first window.</td>
</tr>
<tr>
<td>0.5</td>
<td>Manual fusion method</td>
<td>In this version, the manual fusion method based on the columns the user has selected has been created and connected to the manual fusion window.</td>
</tr>
<tr>
<td>0.5</td>
<td>Save window and Finish window</td>
<td>In this version, the final window of the GUI and its functionalities has been added and connected to the previous window. This is the last version in the first iteration, which is the minimum requirement for this project.</td>
</tr>
</tbody>
</table>

Table 7 the tool versions

4.3 The GUI Implementation

Each window in the GUI has been implemented by using its own class and set of methods with different functionalities. The next section describes the functionality of each class and provides a list of its methods and their descriptions.
• **FusionToolUI Class**

This is the main class and the entry point of the tool that will run the main interface of the AFT. It is responsible for allowing users to select two datasets and to choose the fusion method. The list of methods in this class and their descriptions are in Table 8.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupButton</td>
<td>This method makes the radio buttons used for selecting the fusion method a group, which allows the user to select just one of the methods not both.</td>
</tr>
<tr>
<td>initJRadio</td>
<td>This function makes the Automatic fusion method the default selection.</td>
</tr>
<tr>
<td>BrowseFileActionPerformed</td>
<td>This method executes when the user presses the Browse button in the first window. It opens the Open file window to allow the user to select a CSV file from the home directory and then load the file to the main window.</td>
</tr>
</tbody>
</table>
| NextActionPerformed            | This method executes when the user presses the Next button in the first window. It is responsible for:  
  1. Reading the two CSV files (the datasets) by calling the ReadCSV method from the ReadingFiles Class, which is described later in this section.  
  2. Generating error message 1 if the user has not selected one of the files. The list of error messages is described later in this chapter.  
  3. Checking whether the predefined keys the user selected from the first window exist in the datasets. Otherwise, error message 2 pops up for the user. |
| CancelButtonActionPerformed    | This method closes the tool if the user presses the Cancel button.                                                      |

Table 8 list of methods in “FusionToolUI” Class

• **NextFrame Class**

This class is for creating the window where the user selects the columns from both datasets to include in the generated CSV file. It is responsible for passing the selected columns to the fusion method where the two datasets are mapped and combined in one single dataset. The list of methods in this class is described in Table 9.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formWindowOpened</td>
<td>This method shows the headers of the columns from each dataset as a list of check boxes.</td>
</tr>
<tr>
<td>ActionHandler</td>
<td>This method controls the select all check boxes that is under each list of columns for each dataset.</td>
</tr>
</tbody>
</table>
BackButtonActionPerformed  This method returns the user to the previous window if the user presses the Back button.

CancelButtonActionPerformed  This method closes the tool if the user presses the Cancel button.

BtNextActionPerformed  This method executes when the user presses the Next button. It calls the FindSelectedFields method to start the merging process.

FindSelectedFields  This method retrieves the columns the user has selected and passes them to the suitable fusion algorithm based on the user selection. These fusion algorithms are DataFusion or FusionOfGP_CCGData from Automatic_Fusion class, which is described later in this section.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackButtonActionPerformed</td>
<td>This method returns the user to the previous window if the user presses the Back button.</td>
</tr>
<tr>
<td>CancelButtonActionPerformed</td>
<td>This method closes the tool if the user presses the Cancel button.</td>
</tr>
<tr>
<td>NextButtonActionPerfomed</td>
<td>This method executes when the user presses the Next button from the window. It retrieves the columns the user has selected from each dataset and passes them to the CheckMatchingRecords method. In the case where the user hasn’t selected a column from one of each dataset, error message 3 pops up.</td>
</tr>
<tr>
<td>CheckMatchingRecords</td>
<td>This method checks whether there is a match between the two columns the user has selected. The decision of whether the two columns are matching is based on the number of similarities between the values in each column. If the similarity between them is high (based on a particular threshold), the decision of the ability to merge the datasets will be yes and the user will move to the next step in merging the files. Otherwise, an error message pops up to tell the user to choose another column (error message 4).</td>
</tr>
</tbody>
</table>

Table 9 list of methods in “NextFrame” Class

- ManualFrame Class

This class is for creating the window of the manual fusion method where the user has to select a common column in both datasets to use in matching across the datasets. It allows the user to select the common column from the datasets and directs the user to the next window, which is the NextFrame window that is described above. The list of methods in this class is described in Table 10.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackButtonActionPerformed</td>
<td>This method returns the user to the previous window if the user presses the Back button.</td>
</tr>
<tr>
<td>CancelButtonActionPerformed</td>
<td>This method closes the tool if the user presses the Cancel button.</td>
</tr>
<tr>
<td>NextButtonActionPerformed</td>
<td>This method executes when the user presses the Next button from the window. It retrieves the columns the user has selected from each dataset and passes them to the CheckMatchingRecords method. In the case where the user hasn’t selected a column from one of each dataset, error message 3 pops up.</td>
</tr>
<tr>
<td>CheckMatchingRecords</td>
<td>This method checks whether there is a match between the two columns the user has selected. The decision of whether the two columns are matching is based on the number of similarities between the values in each column. If the similarity between them is high (based on a particular threshold), the decision of the ability to merge the datasets will be yes and the user will move to the next step in merging the files. Otherwise, an error message pops up to tell the user to choose another column (error message 4).</td>
</tr>
</tbody>
</table>

Table 10 list of methods in “ManualFrame” Class
• **FinishForm Class**

This class is to create the last window in the tool. It simply confirms that the result dataset has been generated and gives the user the option to finish and close the tool or use the tool again. The list of methods in this class is described in Table 11.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FinishButtonActionPerformed</td>
<td>This method closes the tool when the user presses the <strong>Finish</strong> button.</td>
</tr>
<tr>
<td>NewButtonActionPerformed</td>
<td>This method allows the user to use the tool again when the user presses the <strong>New</strong> button, which redirects the user to the first window to start a new data fusion process.</td>
</tr>
</tbody>
</table>

Table 11 list of methods in “FinishFrame” Class

**4.4 Back-end Implementation**

This section describes all the classes in the back end that have been implemented to deal with the data passed from the user interface. These classes process the data and return the results to the user interface to show it to the user. In addition, this section describes a configuration file containing a list of the GPs in England with their CCG that is used for mapping the datasets.

• **ReadingFiles Class**

This class deals with the CSV files. It has a single method, which is described in Table 12.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>readCSVFile</td>
<td>This method reads a CSV file and returns it as a list of arrays so the other methods can easily deal with each line in the file as an array of data. It takes the file name as a parameter.</td>
</tr>
</tbody>
</table>

Table 12 method in “ReadingFiles” Class

• **Datasets Class**

This class represents the datasets. It is used mainly to save the datasets and the important parts of each dataset such as the header fields and the index number of the key column in each dataset.
• **Check_existing_keys Class**

This class includes methods to check whether the key identifiers, namely GP, CCG, or Trust codes the user has selected from the first window, exist in the datasets. Otherwise, an error message pops up to tell the user that the automatic fusion can’t be completed. The list of methods in this class is described in Table 13.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPKeyIdentified</td>
<td>This method searches for the GP code in both datasets.</td>
</tr>
<tr>
<td>CCGKeyIdentified</td>
<td>This method searches for the CCG code in both datasets.</td>
</tr>
<tr>
<td>NHSKeyIdentified</td>
<td>This method searches for the Trust code in both datasets.</td>
</tr>
<tr>
<td>CCGGPKeyIdentified</td>
<td>This method searches for the CCG and GP code in both datasets.</td>
</tr>
</tbody>
</table>

**Table 13 list of methods in “Check_existing_keys” Class**

• **Automatic_Fusion Class**

This class is responsible for merging (fusing) two datasets. It includes two fusion methods with different techniques. The list of methods for this class is described in Table 14.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataFusion</td>
<td>This method merges two datasets by using a deterministic one-to-one matching of the common key columns across the datasets. This method is used for automatic fusion for the same key in both datasets and for manual fusion.</td>
</tr>
<tr>
<td>FusionOfGP_CCGData</td>
<td>This method is responsible for merging two datasets with an implicit relationship of the key columns; for example, to merge a dataset that has a GP code as the key column with another dataset that has a CCG code. The linkage between these keys is deterministic and implicit in the tool. The list of all GPs and their CCGs are saved in the tool as a configuration file. The list of GP and CCG files is described later in this chapter.</td>
</tr>
</tbody>
</table>

**Table 14 list of methods in “Automatic_Fusion” Class**

• **generateFiles Class**

This class is responsible for generating and saving the result CSV file by displaying a Save as dialog box which allows the user to choose a folder to save the file in. It has a single method which is described in Table 15.
<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>generateCSVFile</td>
<td>This method generates the result CSV file and displays a Save as dialog box for the user to select a folder to save the file in.</td>
</tr>
</tbody>
</table>

Table 15 method in “Automatic_Fusion” Class

- **GPCCGList file**

  This source file that is used in the automatic fusion of datasets contains GP and CCG codes as the key identifiers. This is a CSV file that contains a list of the 8370 GPs in England with their postcode and the CCG code which organises the work of each GP. The list of practices and names is taken from the Organisation Data Service (ODS), which can be found on the ODS web page [18].

  The list has been added to the tool as a configuration file, not as a separated class, to make it easy for the user to update the list. The user can add more GP codes, add new fields that the user thinks will be useful for merging other healthcare datasets, or even add a new CSV file as a configuration file with data which can be used to map datasets that have not been considered in this project.

  In **FusionOfGP_CCGData method**, the file was used to map the GP code in one dataset with the CCG code in the other dataset. This is a one-to-many matching as each CCG is responsible for organising the work of a number of GPs in a geographic area. A description of the automatic fusion of datasets by using the GPCCGList source file is illustrated in the flowchart in Figure 14.
4.5 Error Messages

Error messages are displayed to inform the user when an unexpected situation happens or the user has gone wrong. This type of message should be informative not encrypted and use terms that are familiar so the user can understand them. One of the characteristics of good software is generating helpful error messages that give the user constructive advice on the way the user can fix the problem [19].

In AFT, error messages have been created to provide direct feedback for the wrong actions the user might take. The list of cases where the tool shows error messages follows.

- **In the first window**

1. When the user presses the **Next** button without selecting any of the datasets, the following error dialog box displays:
The message specifies which file is missing for the first or the second dataset.

2. When the selected identified key, in the automatic fusion method, does not exist in the datasets, the following error dialog box displays:

![Error message for not finding the selected key](image)

The message specifies that the selected key is not found in a specific dataset. In addition, it gives the user a hint to choose another key or to use the manual fusion method to select the column to use for the data fusion.

3. When the user wants to close the tool and presses the Close button, the following dialog box displays:

![Confirmation message before closing the tool](image)
In the manual fusion window
The user must select a column in each dataset to match records across the datasets. Therefore, if the user hasn’t selected a column in one of the datasets, the following error message displays:

Figure 18 Error message for not selecting a column in the manual fusion

4.6 Testing
Software testing is an iterative process that is carried out during and after the implementation. This investigation process has two different purposes. One purpose is to find faults or defects in the software that makes the software behaviour undesirable or gives incorrect results [13]. The second purpose of testing is to demonstrate that the software meets its requirements [13]. This means that all the user requirements must be present in the system and must be tested to check that they work properly. Thus, testing is the process conducted to “build confidence in the software” [13].

Testing software can be performed in different levels with various methods. For this project, the developed tool was tested by using two different methodologies. The first was component testing, also called unit testing, where the individual components were tested individually. Then, an incremental integration testing was used where the software was tested as two or more components integrated together during the development.

4.6.1 Component Testing
The main goal of component or unit testing is to test the smallest testable part of software in isolation from the other parts in the application to examine its behaviour. Unit testing was carried out during the entire implementation process of the project. The Test plane was used to test each version of the software as it is mentioned in section 4.2.

4.6.1.1 GUI testing
Each window in the user interface was tested before creating the next window. This included testing all the actions the user has to perform in the window and its response, not including the interaction between
the interface and the back-end methods. For example, the first window was tested to check that the
window components such as buttons, lists, and the radio buttons respond as expected.

### 4.6.1.2 Back-end Methods Testing

Each class and its methods that were developed as a part of the back-end subsystem were tested
individually with an input test data. Table 16 lists all the tests that have been carried out for each window
in the user interface as well as the back-end methods. Screenshots of testing results and an example of the
generated dataset attached in appendix G.

<table>
<thead>
<tr>
<th>Window 1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionality</strong></td>
<td><strong>Action</strong></td>
<td><strong>Output</strong></td>
</tr>
<tr>
<td>1. <strong>Select the first dataset</strong></td>
<td>Press the <strong>Browse</strong> button and select a file from the open file dialog box.</td>
<td>The selected file is loaded to the tool and the full path is show in the file txt fields.</td>
</tr>
<tr>
<td>2. <strong>Select the second dataset</strong></td>
<td>Same as above</td>
<td>Same as above</td>
</tr>
<tr>
<td>3. <strong>Select key</strong></td>
<td>Press arrow in the key list.</td>
<td>The list of predefined keys opens.</td>
</tr>
<tr>
<td>4. <strong>Close the tool</strong></td>
<td>Press the <strong>Close</strong> button.</td>
<td>A dialog box opens to confirm that the user wants to close the tool.</td>
</tr>
<tr>
<td>5. <strong>Move to the next window</strong></td>
<td>Press the <strong>Next</strong> button.</td>
<td>The next window opens, which depends on the selected fusion method.</td>
</tr>
</tbody>
</table>

**Window 2 (the manual fusion window)**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6. <strong>Back to the previous window</strong></td>
<td>Press the <strong>Back</strong> button.</td>
<td>User returns to the previous window.</td>
</tr>
<tr>
<td>7. <strong>Close the tool</strong></td>
<td>Press the <strong>Close</strong> button.</td>
<td>A dialog box opens to confirm that the user wants to close the tool.</td>
</tr>
<tr>
<td>8. <strong>Move to the next window</strong></td>
<td>Press the <strong>Next</strong> button.</td>
<td>The next window opens for user to select the columns to include in the generated dataset.</td>
</tr>
</tbody>
</table>

**Window 3 (to select column to include in the generated dataset)**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9. <strong>Remove select all option</strong></td>
<td>Remove the check</td>
<td>The check no longer</td>
</tr>
<tr>
<td>10. Back to the previous window</td>
<td>Press the <strong>Back</strong> button.</td>
<td>User returns to the previous window.</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>11. Close the tool</td>
<td>Press the <strong>Close</strong> button.</td>
<td>A dialog box opens to confirm that the user wants to close the tool.</td>
</tr>
<tr>
<td>12. Move to the next window</td>
<td>Press the <strong>Next</strong> button.</td>
<td>The Save as window opens for user to save the new merging file.</td>
</tr>
</tbody>
</table>

**Finish window**

| 13. Close the tool            | Press the **Finish** button. | The tool closes. | No error |
| 14. Start using the tool again| Press the **New** button.    | User returns to the first window. | No error |

**Back-end methods**

| 15. Reading CSV file          | Press the **Next** button in the first window. | The file is read and the list of columns in each dataset appears in the next window. | No error |
| 16. Check whether the selected key exists | Press the **Next** button in the first window. | User receives an error message from the system when one of the keys does not exist in the datasets. | No error |
| 17. Automatic Fusion for similar key in both datasets | Press the **Next** button in the third window. | The Save as window opens, and the user can open the file generated from merging the data. | No error |
| 18. Automatic Fusion for GP dataset with CCG dataset | Press the **Next** button in the third window. | The Save as window opens, and the user can open the file generated from merging the data. | No error |
| 19. Manual Fusion             | Press the **Next** button in the third window. | The Save as window opens, and the user can open the file generated | No error |
Table 16 list of tests and their results

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>20. Generate CSV file</td>
<td>Press the Next button in the third window.</td>
<td>The user can open the new generated file.</td>
</tr>
</tbody>
</table>

Test 17 initially raised for some of the test datasets an out of bounds exception error which occurred when
the code tried to access an array with an index that exceeded its length. This error occurred because some
of the datasets that are downloaded from the HSCIC website are not consistent and contain noise. As the
HSCIC datasets are published monthly, some of these datasets contains empty fields or even have
redundant fields, so the tool can’t get the exact number of fields. Thus, the test datasets have been
changed to ensure they are clear from noise and consistent.

4.6.2 System Testing

System testing is the process of integrating two or more system components and testing their interactions.
This kind of test helps to detect problems that arise from the full system component interactions. System
integration involves adding some code to make the different components of a system interact and work
together[13]. This testing has been done in this project by integrating the GUI and the back-end classes.
Each window of the GUI will get data from the user and pass it to the back-end classes to save or process
the data to get the final result.

The major problem of the integrating test is locating errors [13]. When the error occurs in this stage, the
developer may find it difficult to locate the error, and fixing it is more costly because of the additional
time that is required. To avoid this problem, an incremental integrating testing approach was used in the
system integrating stage of this project. Due to the unit testing that was conducted during the
implementation phase, integrating the GUI with the back-end classes did not reveal any issues to be fixed.

4.7 Deployment

After the automated fusion tool is tested to ensure it is working correctly, the tool has to be deployed so it
can be run on different machines and users can install it and run it on their computers. However, before
doing that, it is necessary to package all the Java class files and the configuration files into a single JAR
(Java ARchive) file.

The JAR [20] file packages all the Java class files, Java form files for the graphical user interface, and the
associated resources such as images and GP-CCG list saved as a configuration CSV file, into a single
executable file with the .jar extension. This allows the entire application to be downloaded in a single request rather than downloading each file separately.

The fusion tool class files and the associated resource files have thus been packaged as a single executable JAR file named Fusiontool. However, the executable file has not been deployed as the file shows an error when trying to run it from the command line. The error is that the configuration file is not found in the same path as the jar file, although the file was already allocated in the jar file. Several attempts were made to fix this problem, however, these failed. Therefore, it was decided not to deploy the tool and run the evaluation on the same machine that the tool was developed on.

4.8 Chapter Summary

Overall, the chapter first explains the implementation phase of the project, including the development tool, the different versions of the tool, and the implementation of the GUI and the back end of the system. It contains a description of all the classes and methods created for each subsystem. Further the chapter also provides the different cases where the system generates error messages for users with screenshots of the error dialog boxes. The final section describes different system tests that have been applied for this project.
Chapter 5. Evaluation

5.1 Technical evaluation

Technical evaluation for the tool was conducted to evaluate two quality-of-service (QoS) considerations which are: performance and scalability. Performance evaluation investigates how the developed tool performs in terms of responsiveness under specific load, namely, dealing with large quantities of records in the provided datasets. This refers to the elapsed time, which is the time taken by the system to respond to the user request for merging two datasets and generating the output file. On the other hand, scalability evaluation investigates the effect of dataset size on the system response time. In addition, in the performance and scalability evaluation, the memory usage was analysed for the tool.

To measure the performance and the scalability of the tool, the developer has to answer the following questions:

- Is the response time fast enough to generate the required output?
- Does the size of the datasets affect the system response time?

5.1.1 Method

The evaluation is divided into three parts. The first analyses the amount of time the system takes to perform the fusion and generate the result file, which is the performance evaluation. The second investigates the effect of the size of the datasets on the response time, which is the scalability of the tool. The last part, analyses the memory usage of the tool.

The tool was implemented in Java using NetBeans IDE. The evaluation was run on Leeds University School of Computing machines, running Fedora 64-bit Linux, with core(TM) i5-2400 3.10GHz Intel processor and 15.5GB RAM.

The data used in the evaluation were six CSV files of health-care datasets downloaded from the HSCIC website. These datasets contain data from GPs, CCGs, and the Cancer Network that can be used to perform the automatic and manual fusion methods. The datasets had from 149 records (in Bowel_Cancer_Audit.CSV) to 41913 records (in prescribing_data_for_CCG.CSV). A list of datasets used in the evaluation, with full details about each set of data is in table 17. For the scalability evaluation, three versions of each dataset have been created, each with a different number of records. More details about the size of the datasets are in the scalability section below.
The NetBeans profiling tool has been used to monitor CPU performance of the system and the memory usage. For each fusion method, a profile process has been performed by running the system, importing two test datasets, and completing all the steps to fuse the datasets. The results displays information on the time spent in each method. Figure 19 illustrates an example of the profiling tool’s result window.
5.1.2 Results

5.1.2.1 Performance

For the performance evaluation, the profiling tool has been used to record the information about the time in three cases. The first case is to use the tool to perform automatic fusion for two datasets with a common key (GP Code). The second case is to perform automatic fusion between a dataset with a GP code as a key identifier and another dataset with a CCG code as a key. The last case is to use the fusion tool to perform manual fusion for two datasets, where the user has to select manually the common key in the datasets. In all three cases, all the columns from both datasets have been selected to be included in the generated dataset. Table 18 shows the performance evaluation of the three cases.

<table>
<thead>
<tr>
<th>Test Name</th>
<th>No. of records in first dataset</th>
<th>No. of records in second dataset</th>
<th>Total time for running all the methods</th>
<th>Time for running the fusion method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic Fusion for two datasets with common key</td>
<td>3051</td>
<td>4412</td>
<td>76313ms</td>
<td>221ms</td>
</tr>
<tr>
<td>Automatic Fusion For GP and CCG datasets</td>
<td>3051</td>
<td>214</td>
<td>51149ms</td>
<td>818ms</td>
</tr>
<tr>
<td>Manual Fusion</td>
<td>3051</td>
<td>4412</td>
<td>14655ms</td>
<td>179ms</td>
</tr>
</tbody>
</table>

Table 18 shows the performance evaluation for the three cases.

The total running time, in table 17, is the time taken to run all the required methods to generate the final result. Figure 20 shows the methods that took the highest running time in each evaluation test.
Figure 20: Methods with highest running time

(a) Time taken for running the Automatic Fusion for two datasets with common key:
- generateCSVfile: 73.80%
- Checkmatchingrecords: 15.30%
- initComponents: 8.40%

(b) Total time taken for running the Automatic Fusion for two datasets with GP and CCG code:
- FusionofGP_CCGData: 50.80%
- main: 32.90%
- generateCSVfile: 11.80%

(c) Time taken for running Manual fusion:
- DataFusion: 51.20%
- readCSVFile: 24.80%
- Checkmatchingrecords: 14.50%
- initComponents: 2.20%
- generateCSVfile: 1.70%

Figure 20 methods with highest running time
The results show that in all three cases most of the total running time is spent in generating the resultant CSV file, which contains the integrated dataset. Then, browsing the computer files to select the datasets is the second highest running time, while methods such as Checkmatchingrecords, Datafusion, FusionOfGP_CCGData, initcomponent, readCSVfile, and other methods took the rest of the running time.

Moreover, the results in table 17 show that the total time and the time for running a specific fusion method differ depending on the size of the datasets and on the type of the fusion method. The total time for the first test is higher than the total time taken for the second test due to the size of the second datasets. The number of records in the second dataset used in the second test is much smaller than that of the dataset used in the first test. In contrast, running the fusion method took much more time for fusing GP and CCG datasets, in the second test, than the time for fusing datasets with common key in the first test. This is because fusing a GP dataset with a CCG dataset requires the use of the configuration file “GPCCGList.csv” as a third file to match each GP code with its CCG code.

For the manual fusion method, the total time for running all the methods as well as for running only the manual fusion method is much smaller than the automatic fusion in the first two tests. The main reason for this is that in the manual fusion, the user has to select the common key from both datasets. Therefore, the manual fusion method does not require a search for the common key in both datasets, as is the case in the automatic fusion.

5.1.2.2 Scalability

For the scalability evaluation, three versions of each of the test datasets have been created, each with different number of records. The first version contains double the number of records as in the original dataset. The second version has 4 times the number of records as in the original dataset. The third version contains 8 times the number of records as in the original dataset.

To measure the scalability of the system, the tests which were performed in the performance evaluation have been conducted with the different versions of the datasets. Overall, the time taken by the fusion methods to merge the datasets was gradually scaled with the number of records in the input datasets. Figure 21 illustrates how the executing time of the three fusion methods scales with the number of records.
Figure 21 executing time for fusion method in: (a) automatic fusion for two datasets with common column, (b) automatic fusion for GP and CCG datasets, and (c) manual fusion
5.1.2.3 Memory Usage

The heap size in Java is the memory size which allocates from the operating system to JVM applications which manages the heap for Java applications [21]. For any new object being created by the application, JVM allocates an area of the allocated heap to store it [21]. If the objects being created exceed the heap size of the application, JVM will show an error.

To ensure that the tool is not exceeding the allocated heap size, the size of the heap was analysed in the performance and scalability evaluation. Figure 22 shows the size of the heap allocated for the tool and the maximum used heap for automatic fusion in the performance evaluation using NetBeans profiling tool.

Figure 22 the size of heap and maximum used heap

Figure 23 shows the heap size and maximum used heap when using the tool to automatically fuse datasets with x8 size of the original datasets.
5.1.3 Discussion

The performance evaluation shows that the automated fusion tool takes a reasonable executing time to perform each of the three tests. These three tests were intended to evaluate the different fusion methods in the fusion tools, namely, automatic and manual fusion. Using automatic fusion for fusing a dataset with common column shows that it takes much less time than in the other case, where GP and CCG datasets are automatically fused. This was logically expected due to the use of a third source file to complete the fusion. Overall, the response time for the developed tool can be considered high, which answers the first measurement question, which is in section 5.1.1.

On the other hand, the scalability evaluation shows that data fusion methods’ executing time is increased with input size. The amount of the increases in the executing time can indicate the complexity of the method. From figure 21, it can be seen that the executing time for the automatic and manual fusion methods used in the developed tool seems to increase exponentially as the input data increases.

Finally, analysing the memory usage shows that the heap size used by the tool was within the range of the allocated heap for the entire application and now errors were reported.

5.2 User Evaluation

Testing the developed system with real users is the most substantial usability method. It provides direct feedback about how people interact with the system and identifies the difficulties that may face with “the concrete interface being tested” [22]. Thus, employing a suitable evaluation technique is an important
decision for this project as it helps to find usability error and user interface weaknesses, at an early stage [22].

The user evaluation in this project investigated the usability of the automated fusion tool, based on specific measurements described in the next section. The evaluation method that was used involves evaluating the system by an expert and by observing 3 public users.

5.2.1 Method

Four types of data from the user evaluation were analysed. The first part of the evaluation analysed: (a) the user interface screens are easy and convenient to use, (b) error messages are clear and make it easy to make corrections, and (c) the use of terms throughout the system is consistent. The second part of the evaluation analysed the ease of using the data fusion tool to integrate two datasets based on a common column compared with the same process but using Microsoft Excel. The aim of comparing the tool with widely used software such as Excel was not to prove that our tool is more efficient or powerful. It was just to show that fusion datasets can be done efficiently using Excel; however, it needs some sort of knowledge, complicated process, and long-time especially if the datasets contains thousands number of fields and records.

5.2.1.1 Participants and materials

Four individuals took part in the user evaluation. One of the users was Dr Roy Ruddle, the project supervisor. The other 3 individuals were master’s students with different backgrounds.

The evaluation took place on Leeds University School of Computing machines, running Fedora Linux. An evaluation form contained multiple tasks and test datasets were provided to the users for use in the evaluation. Each user took approximately 15 minutes to complete the evaluation.

The evaluation form was designed to provide a brief description about the tool to be evaluated, four tasks which cover all the tool use cases to be performed in the first part of the evaluation, and a task for the second part of the evaluation. Each task was prepared to take a specific measurement (as in section 5.2.2.1).

The first part of the evaluation included four tasks. The first task was aimed to measure if error messages in the tool are clear and make it easy for the user to make corrections. The task comprised performing 5 steps, using the tool. Performing these steps caused different error messages to pop up for the users. After each error message, the user has to answer a question, asking if he/she understands the error message.
The next three tasks were aimed to measure whether the user interface screens are easy and convenient to use as well as whether the use of terms throughout the system are consistent. Each of these tasks was designed to ask the user to perform a different step.

In task 2, the user was asked to use the tool to perform an automatic fusion of two datasets provided by General Practices (GPs) in the UK. Both datasets include GP code as a common key. The task contained 11 steps, separated into two parts. In the first part, the user was requested to select two specific datasets provided for the user, the General Practice key for each dataset, and the automatic fusion method. In the second part, the user was asked to select specific columns from each dataset to include in the resulting dataset then save the new CSV file in a determined file.

Task 3 was designed to use the tool to perform an automatic fusion of two datasets, with different keys. The first dataset included GP code as a key while the other datasets included CCG code, as a predefined key. In this case, a third source file, containing a list of GP codes and their CCG codes, was used to find matching records between the datasets.

The task contained steps similar to those in the previous task. The user was first asked to select: two datasets, the General Practice for the first key, and the Clinical Commissioning Group in the second dataset, and the automatic fusion method. Then, the user was asked to select all the columns in the first dataset and some specific columns from the second dataset to include in the generated CSV file then finally, save the file.

In task 4, the user was requested to carry out manual fusion of two datasets. The task was separated into three parts, with each part containing steps to be performed in each window in the user interface. In the first window, the user was asked to select two datasets and the manual fusion method. In the second window, the user was requested to select a particular common column in both datasets. Finally, the user was asked to select all the columns from both datasets to include in the new generated file and save the file.

For the second part of the evaluation, the user was asked to use Microsoft Excel to merge two datasets provided for the user in two Excel files. The task included 12 steps to apply a specific function called “VLOOKUP”. First, the user was asked to open two datasets, which were in two separate Excel files, copy the contents of the second datasets into a new sheet of the same file as the first dataset. Then, the user had to apply the “VLOOKUP” function for each column in the second dataset, to return the matching
records and paste them in, with the first dataset. The evaluation form contained the tasks for the first and second part of the evaluation, as can be viewed in appendix D.

5.2.1.2 Procedure
The evaluation was conducted in two steps. In step one, the evaluation was run with Dr Roy Ruddle, the project supervisor, as an expert in software development. The feedback provided by the expert was considered and the system was updated based on that. Then step 2 of the evaluation was run with 3 users. The users’ observation and feedback was collected to carry out further enhancements to the system. Figure 24 illustrates the user evaluation method.

![Figure 24 the user evaluation method](image)

The evaluation was run for each user separately. The users were first informed of the time needed to complete all the tasks. Then, the users were given a sheet of paper with evaluation instructions, the aim of the evaluation and a brief description about the tool.

The users were given each task to perform separately. During the evaluation, the evaluator (for this project, the evaluator was the developer of the tool) was observing the users and writing comments on the observation sheet (attached in appendix D). The time taken for each task was also recorded. Between each task, the evaluator took around 2 minutes to complete writing the observations and prepare for the next task. The evaluator also answered the users’ questions during the evaluation.

5.2.2 Results

5.2.2.1 Step 1 Results
During the evaluation, Dr Ruddle provided the developer with some qualitative feedback related to the user interface, error messages, and the evaluation processes.
The first feedback was about the terms used in the error messages, which have to be consistent with the terms in the user interface, enabling the user to understand where the error has occurred. For example, the term “identifier key” in the error message, which appears when the selected key is not found in the dataset, did not refer to any term in the interface window. Therefore, the term, “identifier key” was replaced by the term, “key” which is exactly the term used in the first window. This amendment was then applied to all other inconsistent terms in the other error messages. Table 19 shows what was changed in the error messages.

<table>
<thead>
<tr>
<th>Previous error message</th>
<th>comments</th>
<th>Error message after changing</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Please choose file1”</td>
<td>The term “file1” does not exist in the first window</td>
<td>“Please choose a file for the first dataset”</td>
</tr>
<tr>
<td>“Can’t complete merging the files. Identified key is not found in File1. Hint: choose manual fusion method to choose another column to use for matching data across the two files”.</td>
<td>The term “file1” and “identified key” is not exist in the first window. Also, give: the name of the key that was not found and another option for the user to change the key first before the suggestion of using manual fusion.</td>
<td>“Can’t complete Automatic fusion for the files; the key general Practice is not found in the first dataset. Hint: choose another key or use the manual fusion method if you want to use another column to fuse the datasets”.</td>
</tr>
<tr>
<td>“Please select a column from the second dataset to use in matching data”</td>
<td>The term “matching” is not the term used in the tool. Replaced with fusion.</td>
<td>“No column is selected from the first dataset. Please be sure to select a column from both datasets to use in the fusion.”</td>
</tr>
</tbody>
</table>

Table 19 Changes in the error messages.

The second feedback was related to the size of the manual fusion window. In this window, the columns in each datasets are set out as a list of radio buttons inside a scroll pane. If the number of columns or the size of the texts is larger than the pane, the user can use the scroll bar to see the rest of the columns. Figure 23 show an example of the scroll pane used.
Therefore, the suggestion was to make the window larger in order to make the columns clear for the user and so that they do not have to scroll down to see all the columns. This suggestion was considered and the required change was made to the size of the window.

**5.2.2.2 Step 2 Results**

The first part of the evaluation was to use the tool to perform four different tasks. In each task, the evaluator made observations related to the user evaluation measurements.

In the first task, which was to analyse the clarity of the error messages, all the users were able to complete the task. The users were able to understand the error messages and immediately performed the correct action after each error. The average time to complete this task was 2 minutes.

For analysing the ease of using the user interface screens and the consistency of the terms on the screens, tasks 2, 3, and 4 were run. In task 2, all users were able to complete the task easily without asking questions or facing difficulties on any of the screens. The users could navigate through the screens and directly select the correct component, as described in the task such as clicking a button or checking a box. The average time to complete this task was 2 minutes.

In task 3, all the users were able to complete it in a very short time, exactly 1 minute. They commented that this task was clear and direct as the steps were similar to the previous steps.

In task 4, users were asked to carry out manual fusion for two datasets. One of the users asked if there was a missing step because the user did not have to select a key for each dataset. The user recommended that it should be clear in the task steps to keep the key lists on the screen in the default value or disable the
lists when the manual method is selected. Overall, all 3 of the users liked the tool. The users gave positive comments about the tool and found it easy to use.

In the second part of the evaluation, the users were asked to use Excel to perform fusion between two datasets, in separate Excel files. Some users were confused by some steps, especially the steps related to the formula. They spent some time understanding what to select for each parameter in the “VLOOKUP” formula and one of the users had to do it twice as they had done it in the wrong way the first time. In addition, two of the users commented that it was a complicated process. Overall, the users took around 3 to 4 minutes to apply the formula for just one column, in a file containing 47 columns.

5.2.3 Discussion

Running the user evaluation with an expert first was very effective. This affected step 2 of the evaluation as all the feedback was positive and no further enhancement was needed.

Results from running the first task showed that errors in the tool were clear and made it easy for the users to make corrections. Results from tasks 2, 3, and 4 indicated that the user interface of the tool was easy and convenient to use as the users were able to successfully complete all the tasks in a short time.

In the second part of the evaluation, the results revealed that using the tool to fuse two datasets in a single file was much easier and more convenient than using Excel to complete the same task. This was proved by the time taken, as the users took less time to complete any task using the tool than the time taken to apply a formula to just one column. In addition, all the users asked questions related to the required steps while they were performing the task. Also, the number of steps needed to fuse two datasets with any size using the tool was fixed, while the number of steps required to fuse datasets using Excel was dependent on the size of the datasets.

5.3 Chapter Summary

The automated fusion tool has been used to perform a data fusion between two health-care datasets and the tool has been evaluated using two evaluation criteria. First, the tool has been technically evaluated to analyse the performance and the scalability of the tool. Then, a user evaluation has been conducted in order to evaluate the usability of the tool. Generally, the tool has shown sufficiently good results from both the technical and user evaluations.
Chapter 6. Conclusion

This chapter summarises what has been done in this project in respect of: the initial aim and objectives, evaluating whether the minimum requirements have been achieved, and the limitation of this project.

6.1 Conclusion

The aim of this project was to develop a tool that enables the fusion between health-care datasets provided by the HSCIC into a single dataset for the purpose of data mining and analysis. In order to carry out the fusion, the software development work was divided into five distinct phases. In the first phase, the health-care system in the UK was extensively searched in order to understand the relationship between the different NHS organisations and the unique identifier code used to represent each NHS department. In addition, different data linkage techniques were examined to determine appropriate techniques to integrate the datasets.

In the second phase, the health-care, along with census and police, datasets have been analysed to find the key identifier in each dataset. These key identifiers enable the mapping between records in the datasets in order to fuse them into a single dataset.

In the third phase, the system was designed to include the software architecture and its two components: the front-end and the back-end. In addition, different scenarios of usage and use cases were considered in order to identify the user and system interaction and the different functionalities of the tool. Finally, the user interface of the tool was developed.

The fourth phase was the implementation of the system. This is where the actual fusion tool has been built. All the classes and methods were created in order to get the final result which is a single file with the fusion data. Following the implementation of the project, a set of software tests was performed.

In the final phase, a technical and user evaluation was conducted to investigate the performance, scalability and usability of the tool. The tool has shown good performance, however, the elapsed time in the scalability evaluation showed an exponential increase. Finally, the user evaluation indicated that the tool is quite easy to understand, efficient and usable.
6.2 Aim and Minimum Requirements

This section addresses how the minimum requirements that were set out for the project have been fulfilled.

1. **To Identify fusion or link techniques that will be suitable for linking multiple datasets e.g. deterministic linkage methodology.**

   Chapter 2 discussed the possible data linkage techniques. The deterministic linkage was chosen as the health-care datasets have clearly identified a unique code for each record. These codes represent the NHS organisations that provide these data such as General Practices (GP), Clinical Commissioning Groups (CCG), and Trusts. In addition, the chapter explained all the possible relationships between the health-care datasets.

2. **Develop a tool that can fuse two user-provided datasets (e.g. HSCIC datasets) in CSV format and produce the resultant single dataset in a flat file text format, given dataset field mappings either contained within the tool or provided by the user.**

   Chapter 3 and 4 discussed the design and the implementation of the fusion tool. Section 3.3 lists all the possible scenarios of usage, which are considered as the first step towards having an initial design of the project. This was followed by a description of full user requirements, software architecture, and the user interface and back-end design. Finally, the implementation of the classes, methods, and source files were detailed in chapter 4.

3. **Develop a graphical user interface (GUI) that allows a user to import two datasets in a CSV format and generate the output.**

   The graphical user interface was explained first in section 3.6 in the design chapter with a list of screens and their initial functionalities. Section 4.3 contained a discussion of the implementation of the GUI including details of all the classes, methods and functionality of each component in the different GUI’s windows. Finally, each window in the user interface was tested separately, using the unit testing method followed by a system test of the entire user interface, after integrating all the windows together.

4. **Allow the user to choose the fusion method, namely, automatic, where the tool will find the way to fuse the datasets or manual, where the user can specify the common column in both datasets.**
The first window of the graphical user interface was designed to let the user select a fusion method. For each fusion method, an algorithm was developed to perform the fusion (details in Section 4.4). For the manual fusion method, an extra window was built to enable the user to choose a common key from the datasets to be used in the fusion method.

### 6.3 Limitations:

1. This project should be seen as being composed of two iterations. The first one was to build a tool that can be used to fuse health-care datasets from a single provider, namely the HSCIC. The second iteration was to enhance the tool to fuse the health-care datasets with other datasets such as census and police datasets. However, due to the time constraints, the second iteration has not been conducted.
2. The developed tool has not been deployed because of an error in running the executable file. Thus, the evaluation was run in the developer profile in Leeds University School of Computing machines.

### 6.4 Challenges:

1. The first challenge in this project was in analysing the health-care datasets that required an understanding of the health-care system in the UK and the different unique codes used to identify different NHS organisations.
2. Designing the tool using NetBeans IDE, which was a totally new development environment for the developer, was achieved by spending a lot of time on online tutorials to understand how to use its features and its wide range of tools.
3. One of the main challenges was the implementation of the tool. The tool was developed in Java, with which programming language the user was unfamiliar. Thus, a lot of time was spent learning Java, and developing and testing the code and its functionality.
References


Appendix A
A.1 Project Reflection

Undertaking a big task such as Msc project was a long journey, full of challenges and difficulties that provided me with a broad range of knowledge and skills. One of the main challenges that I faced on this project was to have a clear understanding of the main problem that needed to be solved. The term Data fusion which was in the title of the project was the main confusion for me. After searching in this field, all the resources that were founded related to data fusion were about sensors or medical images data fusion which was totally different than what the project was about. This was where the frequent meeting with my supervisor Dr. Roy Ruddle were useful as he was kept guiding me into the right direction and gave me the chance to explore and find the answered for my questions. By this way I was developing my self-learning skills which I didn’t learn it from my undergraduate studies.

Moreover, the time management during the project period was a difficult challenge for me. Managing the time effectively and making a good plane for the project is considered for me the key success factor to successfully complete the project. However, this was very difficult for me as I am a mom of two children and I was trying to divide my time between the project and my family. At the end, I was able to manage my time, finish the project report on time, and develop my time management skills.

In addition, writing the project report was a real challenge for most of the MSc students. During the year of the MSc study, I realised that the best time to write a report is when you are doing the task that you want to write about. This makes writing the final report much easier and less stressful

Finally, I have really enjoyed working in this project and have the opportunity to gain knowledge and experience and develop my personal skills through the project.
Appendix B

B.1 Materials used in the solution

A number of health care datasets that have been published by the HSCIC under the terms of the open government licence have been used as test data in this project. This data is being made available as comma-separated values (CSV) files in the HSCIC website: [http://www.hscic.gov.uk/searchcatalogue](http://www.hscic.gov.uk/searchcatalogue)

The list of datasets used in the test and the evaluation of this project is:

1. Number of patients registered at GP Practice By GP.
2. Number of patients registered at GP Practice By CCG.
3. Number of patients registered at GP Practice By LSOA.
4. GP practice Prescribing Presentation level.
5. GP practice Prescribing Chemical Level.
6. Index of Multiple Deprivations (IMD).

To map GP codes with CCG in the datasets, a list of the CCG organisational codes and their GPs was used in the developed tool as a source file added to the JAR file of the tool. The list is available from: [www.england.nhs.uk/wp-content/uploads/.../list-of-proposed-practices.xls](http://www.england.nhs.uk/wp-content/uploads/.../list-of-proposed-practices.xls)
Appendix C
C.1 Ethical Issues

In the user evaluation, the participants’ personal or identifiable information has not been collected (The form used in the evaluation is in appendix D). Therefore, this project is out of ethical issues.
Instructions:

- Please complete the following Tasks

<table>
<thead>
<tr>
<th>Task Number</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Note: This evaluation form consists of five printed papers to help MSc student to evaluate a developed tool called “Automated Fusion Tool” for her MSc Final project. It will be used to measure:

1. Accuracy of tool outputs.
2. Screens are easy and convenient to use.
3. Error messages are clear and make it easy to make corrections.
4. The use of terms throughout the system is consistent.
Overview

Automated Healthcare Data Fusion Tool is developed to enable the fusion of two healthcare datasets into a single dataset for the purposes of data mining and analysis. The tool enables the user to perform the fusion in two different methods namely automatically or manually.

In the automatic method the user has to select two datasets, identified keys for each data, and columns from both dataset to include in the generated dataset. The fusion will be performed automatically by the system if applicable.

In the manual fusion method, the user has to select two dataset, a common column from both datasets to use in matching records across the datasets, and columns from both dataset to include in the generated dataset.
**Task 1**

The aim of this task is to measure if error messages in the tool are clear and make it easy for the user to make corrections.

**Steps:**

In the first window, do the following:

1. Press **Next** button. What do you understand from the error message?
2. For the First File, press **Browse** button. Select file (A) and then select (GP1.CSV). Do the same steps for the second file but select (CCG1.CSV).
3. Then press **Next**. What do you understand from the error message?
4. In the same window, select Manual fusion method and press **Next**.
5. Then press **Next**. What do you understand from the error message?
**Task 2**

In this task, you will use the tool to do automatic Fusion of two datasets provided by General Practices in the UK. Both datasets includes general practice (GP) code as a common key column.

**Steps:**

In the first window do the following:

1. For the First File, press **Browse** button. Select the file (A) and then select (GP1.CSV).
2. From **Key** List select General Practice.
3. For the Second File, press **Browse** button. Select the file (A) and then select (GP2.CSV).
4. From **Key** List select General Practice.
5. Make sure that Automatic radio button in the fusion method is selected.
6. Press **Next**

In the Second window do the following:

1. Remove the check from select all check box under the first dataset, and then select the first five columns.
2. Remove the check from select all check box under the second dataset, and then select the first five columns.
3. Press **Next**.
4. In the Save As window select file (B) to save your new file.
5. Then press **New** in (Step 3) window to start the next task.
**Task 3**

In this task, you will use the tool to do automatic Fusion of two datasets. The first dataset includes data provided by General Practices (GP) in the UK and uses General Practice code as a Key. The second dataset includes data provided by Clinical Commissioning Groups (CCG) in the UK and uses CCG Code as a key.

**Steps:**

In the first window, do the following:

1. For the First File, press **Browse** button. Select the file (A) and then select (GP1.CSV).
2. From **Key** List select general Practice.
3. For the Second File, press **Browse** button. Select the file (A) and then select (CCG2.CSV).
4. From **Key** List select Clinical Commissioning Group.
5. Make sure that Automatic radio button in the fusion method is selected.
6. Press **Next**

In the Second Window, do the following:

1. Keep all columns in the first dataset selected.
2. In the second dataset, remove the check from select all check box and select the first four columns.
3. Press **Next**.
4. In the Save As window select file (B) to save your new file.
5. Then press **New** in (Step 3) window to start the next task.
**Task 4**

In this task, you will use the tool to do manual Fusion of two datasets. In this method, a user can choose which column to use for matching data across the two datasets.

**Steps:**

In the first window, do the following:

1. For the First File, press **Browse** button. Select the file (A) and then select the file (Bowel_cancer_audit).
2. For the Second File, press **Browse** button. Select the file (A) and then select the file (Trustcode1).
4. Press **Next**.

In the second window, do the following:

1. Choose which column to use for matching data across the two datasets by selecting the common column (Cancer Network/ Trust Name) from the first file and (Cancer Network) from the second file.
2. Press **Next**.

In the Third Window, do the following:

1. Keep all the columns in both datasets selected then press **Next**.
2. In the Save As window select file (B) to save your new file.
3. Then press Finish.
**Task 5**

The aim of this task is to compare the usability of the AFT with other software such as LibreOffice Calc or Excel that can be used to merge two datasets. In this task, you will use Excel.

**Steps:**

1. From Desktop, open (Evaluation) folder, then open the files (GP1) and (GP2).
2. In GP1, press the (+) icon after sheet1 tab to open new sheet.
3. Copy the content of GP2, by clicking on the cell at the top left corner of the sheet between cell A and 1, and paste it in sheet2 that you have just opened in the previous step.
4. Copy the headers of the columns from sheet2, excluding the first column which is the key column (GP_PRACTICE_CODE), and past them in the empty cells in sheet1. That means cell H1 should has POSTCODE and cell I1 has CCG_CODE etc.
5. Select the first cell under POSTCODE column which is H2, and then select the function wizards from the Formula bar with icon 📊.
6. Select VLOOKUP from the list of functions then press Next.
7. For the search criterion select the first cell under Practice code (E2). You can see that E2 is written in the search criterion field.
8. For the array field, go to sheet 2 and select all the values in the table starting from cell A2 to Z216. After that you can see that (Sheet2.A2:Z216) written in the array field.
9. Index field is the column number in the second datasets from which the matching value must be returned. For Postcode column the index will be 2.
10. Sort order, is a logical value that specifies whether you want VLOOKUP to find an exact match or an approximate match, in this case just write (0). So the formula should be:

   \[=VLOOKUP(E2,Sheet2.A2:Z216,2,0)\]

11. Press OK.
12. Select the previous cell (E2) with the returned value, place your mouse cursor at the lower right-hand corner of the cell. You’ll know you have hit it when the cursor changes to a plus sign (+) and double click to fill the cells with a copy of the original formula.
13. You have to do the same steps or just copy the formula for each column from sheet2 and change the function parameters. The following is the formulas that should be in the second and third column from sheets.

CCG_CODE :

=VLOOKUP(E2,Sheet2!A2:Z216,3,0)

NHSE_AREA_TEAM_CODE

=VLOOKUP(E2,Sheet2!A2:Z216,4,0)
<table>
<thead>
<tr>
<th>Task 1</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start time:</td>
<td></td>
</tr>
<tr>
<td>End time:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 2</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start time:</td>
<td></td>
</tr>
<tr>
<td>End time:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 3</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start time:</td>
<td></td>
</tr>
<tr>
<td>End time:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 4</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start time:</td>
<td></td>
</tr>
<tr>
<td>End time:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 5</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start time:</td>
<td></td>
</tr>
<tr>
<td>End time:</td>
<td></td>
</tr>
</tbody>
</table>
**Appendix E**

**E.1 Project Plane**

<table>
<thead>
<tr>
<th>Task</th>
<th>Mar-14</th>
<th>Apr-14</th>
<th>May-14</th>
<th>Jun-14</th>
<th>Jul-14</th>
<th>Aug-14</th>
<th>Sep-14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Data analysis</td>
<td>W1</td>
<td>W2</td>
<td>W3</td>
<td>W4</td>
<td>W1</td>
<td>W2</td>
<td>W3</td>
</tr>
<tr>
<td>2 Background reading</td>
<td>W1</td>
<td>W2</td>
<td>W3</td>
<td>W4</td>
<td>W1</td>
<td>W2</td>
<td>W3</td>
</tr>
<tr>
<td>3 Interim report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 First iteration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.a Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.b Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.c Evaluation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 Second iteration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.a Linking with non-health datasets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.b Evaluation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 Writing-up final report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Current time*

**Milestones:**

- 20-06-2014: interim report
- 01-08-2014: Progress meeting
- 04-09-2014: final report submission
Appendix F
F.1 Presentation at progress meeting

Background to the problem
- The past decade has seen an increase in the number of published health
  and social care datasets. However, these datasets are kept
  isolated and therefore difficult to use effectively.
- The aim of this project is to overcome this barrier by developing a tool
  that can link health and social care datasets.
- The tool needs to be able to handle multiple datasets and data
  sources.
- The tool needs to be able to handle multiple datasets and data
  sources.
- The tool needs to be able to handle multiple datasets and data
  sources.

Aim and Objectives
- The main objective of this project is to develop a tool that can link
  health and social care datasets.
- The tool needs to be able to handle multiple datasets and data
  sources.
- The tool needs to be able to handle multiple datasets and data
  sources.
- The tool needs to be able to handle multiple datasets and data
  sources.

Minimum Requirements and deliverables
- To identify features and techniques that will be suitable for linking
  across multiple datasets and data sources.
- To develop a tool that can handle multiple datasets and data
  sources.
- To develop a tool that can handle multiple datasets and data
  sources.
- To develop a tool that can handle multiple datasets and data
  sources.

Further enhancements
- To improve the tool’s performance.
- To add new features.
- To add new features.
- To add new features.

Project Methodology
- The project is divided into the following stages:
  1. Data collection and preparation.
  2. Data analysis.
  3. Design.

Data Analysis
- Data collected from the health and social care datasets.
- Data collected from the health and social care datasets.
- Data collected from the health and social care datasets.
- Data collected from the health and social care datasets.

Design
- The design is divided into the following stages:
  2. Detailed Design.
  3. Hardware Design.

Census & Health care datasets

Police & Health care datasets

1. Requirements

2. Data Analysis

3. Design
4. Implementation
- The application is implementing using two solid database development tools.

5. Evaluation
- Technical evaluation which tests the software performance and reliability of the tool.
- User evaluation to test the user acceptance of the tool.

5.1 Technical Evaluation
- Performance evaluation:
  1. Simulate the system to test the test performance in terms of response time under specified load.
  2. Evaluate the tool in divided into two parts:
  3. The first calculates the rate of time the system is required to respond to the user request, including the generation time, the I/O load, and any other external load to the system, whether it is considered to be the response time.
  4. The second investigates the effect of the size of the data set on the response time.

5.2 User Evaluation
- This type of evaluation involves examining whether the user performs more or less using the tool than usual and provides them with various examples.
- Expert review: The application will be examined by three experts.

- Requirements:
  1. Ease of learning and using the application.
  2. Amount of information revelation.
  3. Results of the application's simulations.
  4. Relative error and conversion rate.
  5. Development time and actual time in real-time conversion.
  6. The use of tools throughout the system is consistent.
Appendix G
G.1 Screenshots of Testing

1. The main window of the tool:

2. Selecting two datasets:
3. After importing two datasets and selecting the key for each dataset:

![Automated Fusion Tool](image1.png)

4. Select All the columns from both datasets to include in the generated dataset (the default case in this window):

![Automated Fusion Tool](image2.png)
5. Select some of the columns to include in the generated dataset:

6. Final screen for closing the tool or using the tool again:

Press:

1. Finish: to close the tool.
2. New: to use the tool again.
7. Selecting a common column from the datasets to use in the fusion:
## Example of the Generated CSV:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GP_FRAIPOSTCODE</td>
<td>DOG_CODE</td>
<td>WISE_ARCVISE_RESHA code</td>
<td>SHA</td>
<td>name</td>
<td>PCT code</td>
<td>PCT name</td>
<td>Practice name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A0101</td>
<td>TS15 1HU</td>
<td>INK</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Stockton-on-Tees Teaching</td>
<td>THE DENSHAM SURGERY</td>
</tr>
<tr>
<td>3</td>
<td>A0102</td>
<td>TS15 2AW</td>
<td>INK</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Stockton-on-Tees Teaching</td>
<td>QUEENS PARK MEDICAL CENTRE</td>
</tr>
<tr>
<td>4</td>
<td>A0103</td>
<td>TS26 0DE</td>
<td>INK</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Hartlepool</td>
<td>THE GALLACHER PRACTICE</td>
</tr>
<tr>
<td>5</td>
<td>A0104</td>
<td>TS15 2EE</td>
<td>INM</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Middlesbrough</td>
<td>WOODLANDS ROAD SURGERY</td>
</tr>
<tr>
<td>6</td>
<td>A0105</td>
<td>TS14 7DU</td>
<td>INM</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Retford and Cleveland</td>
<td>SPRINGWOOD SURGERY</td>
</tr>
<tr>
<td>7</td>
<td>A0106</td>
<td>TS15 2AT</td>
<td>INK</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Stockton-on-Tees Teaching</td>
<td>TEMPLE ST STREET MEDICAL PRACTICE</td>
</tr>
<tr>
<td>8</td>
<td>A0107</td>
<td>TS24 7PW</td>
<td>INK</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Hartlepool</td>
<td>BANK HOUSE SURGERY</td>
</tr>
<tr>
<td>9</td>
<td>A0108</td>
<td>TS8 7TD</td>
<td>INM</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Retford and Cleveland</td>
<td>ALBERT HOUSE CLINIC</td>
</tr>
<tr>
<td>10</td>
<td>A0109</td>
<td>TS8 0HF</td>
<td>INM</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Middlesbrough</td>
<td>VILLAGE MEDICAL CENTRE</td>
</tr>
<tr>
<td>11</td>
<td>A0111</td>
<td>TS24 7PV</td>
<td>INK</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Hartlepool</td>
<td>CHADWICK PRACTICE</td>
</tr>
<tr>
<td>12</td>
<td>A0112</td>
<td>TS3 6AL</td>
<td>INM</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Middlesbrough</td>
<td>WESTBOURNE MEDICAL CENTRE</td>
</tr>
<tr>
<td>13</td>
<td>A0113</td>
<td>TS12 2FF</td>
<td>INM</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Retford and Cleveland</td>
<td>BROTTON SURGERY</td>
</tr>
<tr>
<td>14</td>
<td>A0114</td>
<td>TS23 2LA</td>
<td>INK</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Stockton-on-Tees Teaching</td>
<td>QUENSTREE PRACTICE</td>
</tr>
<tr>
<td>15</td>
<td>A0115</td>
<td>TS10 1ZQ</td>
<td>INM</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Retford and Cleveland</td>
<td>LAGAN SURGERY</td>
</tr>
<tr>
<td>16</td>
<td>A0116</td>
<td>TS13 QY</td>
<td>INM</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Middlesbrough</td>
<td>PARK SURGERY</td>
</tr>
<tr>
<td>17</td>
<td>A0117</td>
<td>TS17 3EE</td>
<td>INM</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Stockton-on-Tees Teaching</td>
<td>WOODRIDGE PRACTICE</td>
</tr>
<tr>
<td>18</td>
<td>A0118</td>
<td>TS10 4NN</td>
<td>INM</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Retford and Cleveland</td>
<td>BENTLEY MEDICAL PRACTICE</td>
</tr>
<tr>
<td>19</td>
<td>A0119</td>
<td>TS3 6RL</td>
<td>INM</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Middlesbrough</td>
<td>CROSSFELL HEALTH CENTRE</td>
</tr>
<tr>
<td>20</td>
<td>A0120</td>
<td>TS4 3BU</td>
<td>INM</td>
<td>Q45</td>
<td>Y54</td>
<td>Q20</td>
<td>North East Strategic Authority</td>
<td>50</td>
<td>Middlesbrough</td>
<td>MARTINSIDE MEDICAL CENTRE</td>
</tr>
<tr>
<td>21</td>
<td>B01574</td>
<td>HUO 9RM</td>
<td>ICF</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>Yorkshire And The Humber Strategic Health Authority</td>
<td>Hull Teaching</td>
<td>DR. J. JOSEPH'S PRACTICE</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>B01575</td>
<td>HUO 9RX</td>
<td>ICF</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>Yorkshire And The Humber Strategic Health Authority</td>
<td>Hull Teaching</td>
<td>TAK AH</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>B01577</td>
<td>DN22 5EF</td>
<td>ICH</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>Yorkshire And The Humber Strategic Health Authority</td>
<td>Hull Teaching</td>
<td>DRS. CHALMERS &amp; MEIER</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>B01579</td>
<td>HUO 8PE</td>
<td>ICH</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>Yorkshire And The Humber Strategic Health Authority</td>
<td>Hull Teaching</td>
<td>EAST RIDING OF YORKSHIRE SURGERY</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>C0817</td>
<td>PE6 0AL</td>
<td>ICH</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>East Midlands Strategic Authority</td>
<td>Hull Teaching</td>
<td>ABBEY RIVIERA SURGERY</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>C0826</td>
<td>U11 1XP</td>
<td>IMR</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>East Midlands Strategic Authority</td>
<td>Hull Teaching</td>
<td>BRADFORD MEDICAL PRACTICE</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>C0826</td>
<td>Y11 1XJ</td>
<td>IMR</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>East Midlands Strategic Authority</td>
<td>Hull Teaching</td>
<td>WOODSTAFF SURGERY</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>C0826</td>
<td>PE11 1LT</td>
<td>IMR</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>East Midlands Strategic Authority</td>
<td>Hull Teaching</td>
<td>PENNYCADE SURGERY</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>C0826</td>
<td>DN22 3BB</td>
<td>IMR</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>East Midlands Strategic Authority</td>
<td>Hull Teaching</td>
<td>HAWTHORN SURGERY</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>C0829</td>
<td>U10 7ST</td>
<td>IMR</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>East Midlands Strategic Authority</td>
<td>Hull Teaching</td>
<td>TASTREY LODGE SURGERY</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>C0830</td>
<td>B31 1QZ</td>
<td>ICH</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>West Midlands Strategic Health Authority</td>
<td>Hull Teaching</td>
<td>SOUTH BIRMINGHAM NORTHEAST HEALTH CENTRE F</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>C0831</td>
<td>B41 3EP</td>
<td>ICH</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>West Midlands Strategic Health Authority</td>
<td>Hull Teaching</td>
<td>BIRMINGHAM EAST AND NORTH F</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>C0833</td>
<td>B72 7RL</td>
<td>ICH</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>West Midlands Strategic Health Authority</td>
<td>Hull Teaching</td>
<td>THE HARLEQUIN SURGERY</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>C0834</td>
<td>B88 3PU</td>
<td>ICH</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>West Midlands Strategic Health Authority</td>
<td>Hull Teaching</td>
<td>BIRMINGHAM EAST AND NORTH B</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>C0835</td>
<td>B82 3SD</td>
<td>ICH</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>West Midlands Strategic Health Authority</td>
<td>Hull Teaching</td>
<td>OXIMA PRACTICE</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>C0835</td>
<td>B82 QRA</td>
<td>ICH</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>West Midlands Strategic Health Authority</td>
<td>Hull Teaching</td>
<td>SOUTH BIRMINGHAM WOODCUT VALLEY HEALTH CENTRE</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>C0835</td>
<td>B82 QRA</td>
<td>ICH</td>
<td>Q50</td>
<td>Y54</td>
<td>Q20</td>
<td>West Midlands Strategic Health Authority</td>
<td>Hull Teaching</td>
<td>SOUTH BIRMINGHAM DUDLEY PARK MEDICAL CENTRE</td>
<td></td>
</tr>
</tbody>
</table>
## Appendix H

### H.1 JavaDoc

**Package my.FusionTool**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic_Fusion</td>
<td>This class is responsible for merging (fusing) two datasets.</td>
</tr>
<tr>
<td>Check_existing_keys</td>
<td>This class includes methods to check whether the key identifiers, namely GP, CCG, or Trust codes the user has selected from the first window, exist in the datasets.</td>
</tr>
<tr>
<td>Datasets</td>
<td>This class represents the datasets.</td>
</tr>
<tr>
<td>FinishForm</td>
<td>This class represents the last window which is the &quot;Finish&quot; window.</td>
</tr>
<tr>
<td>FusionToolUI</td>
<td>This class represents the first window in the user interface.</td>
</tr>
<tr>
<td>generateFiles</td>
<td>This class is responsible for generating and saving the result CSV file.</td>
</tr>
<tr>
<td>ManualFrame</td>
<td>This class is for creating the window of the manual fusion method where the user has to select a common column in both datasets to use in matching across the datasets.</td>
</tr>
<tr>
<td>NextFrame</td>
<td>This class is for creating the window where the user selects the columns from both datasets to include in the generated CSV file.</td>
</tr>
<tr>
<td>ReadingFiles</td>
<td>This class deals with the CSV files.</td>
</tr>
</tbody>
</table>
Class Automatic_Fusion

java.lang.Object
   my.Fusiontool.Automatic_Fusion

public class Automatic_Fusion
extends java.lang.Object

This class is responsible for merging (fusing) two datasets.

Constructor Summary

Constructors

 Automatic_Fusion()

Method Summary

Methods

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>DataFusion(int No_col_newdataset, int[] indxfields1, int[] indxfields2)</td>
</tr>
<tr>
<td></td>
<td>This method merges two datasets by using a deterministic one-to-one matching of the common key columns across the datasets.</td>
</tr>
<tr>
<td>void</td>
<td>FusionOfGP_CCGData(int No_col_newdataset, int[] indxfields1, int[] indxfields2)</td>
</tr>
<tr>
<td></td>
<td>This method is responsible for merging GP and CCG datasets.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

 Automatic_Fusion()

public Automatic_Fusion()
Method Detail

DataFusion

public void DataFusion(int No_col_newdataset,
        int[] indxfields1,
        int[] indxfields2)

This method merges two datasets by using a deterministic one-to-one matching of the common key columns across the datasets. This method is used for automatic fusion for the same key in both datasets and for manual fusion.

Parameters:

    No_col_newdataset - this is the number of columns that the user selects to be in the new generated CSV file.
    indxfields1 - the indexes of the selected columns in the first dataset
    indxfields2 - the indexes of the selected columns in the second dataset

FusionOfGP_CCGData

public void FusionOfGP_CCGData(int No_col_newdataset,
        int[] indxfields1,
        int[] indxfields2)

This method is responsible for merging GP and CCG datasets.

Parameters:

    No_col_newdataset - this is the number of columns that the user selects to be in the new generated CSV file.
    indxfields1 - the indexes of the selected columns in the first dataset.
    indxfields2 - the indexes of the selected columns in the second dataset.
my.Fusiontool

Class FinishForm

java.lang.Object
  java.awt.Component
    java.awt.Container
    java.awt.Window
      java.awt.Frame
        javax.swing.JFrame
          my.Fusiontool.FinishForm

All Implemented Interfaces:
  java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible,
  javax.swing.RootPaneContainer, javax.swing.WindowConstants

public class FinishForm
  extends javax.swing.JFrame

This class represents the last window which is the "Finish" window

See Also:
  Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JFrame
  javax.swing.JFrame.Accessible JFrame

Nested classes/interfaces inherited from class java.awt.Frame
  java.awt.Frame.Accessible AWTFrame

Nested classes/interfaces inherited from class java.awt.Window
  java.awt.Window.Accessible AWT Window, java.awt.Window.Type

Nested classes/interfaces inherited from class java.awt.Container
  java.awt.Container.Accessible AWT Container

Nested classes/interfaces inherited from class java.awt.Component
  java.awt.Component.Accessible AWT Component,
  java.awt.Component.Flip Buffer Strategy
Field Summary

Fields inherited from class javax.swing.JFrame
accessibleContext, EXIT_ON_CLOSE, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO NOTHING ON CLOSE, HIDE ON CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

Constructors

FinishForm()
Creates new form FinishForm

Method Summary

Methods

static void main(java.lang.String[] args)
The main function of the class

Methods inherited from class javax.swing.JFrame
addEmpl, createRootPane, frameInit, getAccessibleContext, getContentPane, getCloseOperation, getContentPane, getGraphics, getJMenuBar, getLayeredPane, getRootPane, getTransferHandler, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, repaint, getContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setDefaultCloseOperation, setIconImage, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, setTransferHandler, update
Methods inherited from class java.awt.Frame
addNotify, getContentPane, getExtendedState, getFrames, getIconImage,
getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, undecorated,
remove, removeNotify, setBackground, setCursor, setExtendedState,
setMaximizedBounds, setMenuBar, setOpacity, setResizable, setShape, setState,
setTitle, setUndecorated

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener,
addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle,
createBufferStrategy, createBufferStrategy, dispose, setBackground,
getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor,
getFocusOwner, getFocusTraversalKeys, getIconImages, getInputContext, getListeners,
getLocale, getModalExclusionType, getMostRecentFocusOwner, getOpacity,
getOwnershipStatus, getOwner, getOwnerHasFocus, getShape, getToolkit, getType,
getWarningString, getWindowFocusListeners, getWindowListeners, getWindows,
getWindowStateListeners, hide, isActive, isAlwaysOnTop, isAlwaysOnTopSupported,
isAutoRequestFocus, isFocusableWindow, isFocusCycleRoot, isFocused,
isLocationByPlatform, isopaque, isshowing, isValidateHost, pack, paint, postEvent,
processEvent, processWindowFocusEvent, processWindowStateEvent,
removeWindowFocusListener, removeWindowListener, removeWindowStateListener, reshape,
setAlwaysOnTop, setAutoRequestFocus, setBounds, setBounds, setCursor,
setFocusableWindowState, setFocusCycleRoot, setIconImages, setLocation,
setLocationRelativeTo, setMinimumSize, setModalExclusionType, setSize,
setSize, settype, setVisible, show, toBack, toFront

Methods inherited from class java.awt.Container
add, add, add, add, addContainerListener, applyComponentOrientation,
areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt,
findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt,
getComponentCount, getComponents, getComponentZOrder, getContainerListeners,
getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets,
invalid, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicy, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents,
preferenceComponents, processContainerEvent, remove, removeAll,
removeContainerListener, setComponentZOrder, setFocusTraversalKeys,
setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont,
transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener,
addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener,
addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage,
coalesceEvent, contains, contains, createImage, createImage, createVolatileImage,
createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable,
enableEvents, enableInputMethods, firePropertyChange, firePropertyChange,
firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange,
firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange,
baseline, getBaseline, getBounds, getBounds, getChildren, getComponent,
getComponentOrientation, getCursor, getDropTarget, getFocusableComponents, getFocusTraversalKeysEnabled, getFont, getFontMetrics,
getForeground, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners,
getHierarchyListeners, getIgnoreRepaint, getComponent, getInputMethodListeners,
inputMethodRequests, getListeners, getLocation, getLocationRelativeTo,
getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners,
getName, getPeer, getPreferredSize, resize, setBounds, setCursor,
setDoubleBuffered, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner,
isFocusTraversable, isFontSet, isForeground, isLightweight, isMaximumSize,

88
FinishForm

- isMinimumSizeSet
- isPreferredSizeSet
- isValid
- isVisible
- keyDown
- keyUp
- list
- list
- location
- lostFocus
- mouseDown
- mouseDrag
- mouseEnter
- mouseExit
- mouseMove
- mouseUp
- move
- nextFocus
- paintAll
- prepareImage
- prepareImage
- printAll
- processComponentEvent
- processComponentEvent
- processComponentEvent
- processHierarchyBoundsEvent
- processHierarchyEvent
- processInputMethodEvent
- processKeyEvent
- processMouseEvent
- processMouseMotionEvent
- processMouseWheelEvent
- removeComponentListener
- removeFocusListener
- removeHierarchyBoundsListener
- removeHierarchyListener
- removeInputMethodListener
- removeKeyListener
- removeMouseListener
- removeMouseMotionListener
- removeMouseWheelListener
- removePropertyChangeListener
- removePropertyChangeListener
- repaint
- repaint
- repaint
- repaint
- requestFocus
- requestFocus
- requestFocus
- requestFocus
- requestFocus
- requestFocusInWindow
- requestFocusInWindow
- resize
- resize
- revalidate
- setComponentOrientation
- setDropTarget
- setEnabled
- setFocusable
- setFocusTraversalKeysEnabled
- setForeground
- setIgnoreRepaint
- setLocale
- setMaximumSize
- setName
- setPreferredSize
- show
- size
- toString
- transferFocus
- transferFocusBackward
- transferFocusUpCycle

Methods inherited from class java.lang.Object

- clone
- equals
- finalize
- getClass
- hashCode
- notify
- notifyAll
- wait
- wait
- wait

Methods inherited from interface java.awt.MenuContainer

- getFont
- postEvent

Constructor Detail

FinishForm

public FinishForm()

Creates new form FinishForm

Method Detail

main

public static void main(java.lang.String[] args)

The main function of the class

Parameters:

- args - the command line arguments
Class FusionToolUI

java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Frame
          javax.swing.JFrame
            my.FusionTool.FusionToolUI

All Implemented Interfaces:
  java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible,
  javax.swing.RootPaneContainer, javax.swing.WindowConstants

public class FusionToolUI
  extends javax.swing.JFrame

This class represents the first window in the user interface. It allows the user to:
1. Import a CSV file for the first dataset.
2. Select a key for the first dataset.
3. Import a CSV file for the second dataset.
4. Select a key for the second dataset.
5. Select a fusion method namely: Automatic or Manual.

See ALSO:
  Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JFrame
javax.swing.JFrame.AccessibleJFrame

Nested classes/interfaces inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Nested classes/interfaces inherited from class java.awt.Window
java.awt.Window.AccessibleJWTCWindow, java.awt.Window.Type

Nested classes/interfaces inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes/interfaces inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent,
java.awt.Component.FlipBufferStrategy
Field Summary

Fields inherited from class `javax.swing.JFrame`

accessibleContext, EXIT_ON_CLOSE, rootPane, rootPaneCheckingEnabled

Fields inherited from class `java.awt.Frame`

CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_MIRROR, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class `java.awt.Component`

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface `javax.swing.WindowConstants`

DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface `java.awt.image.ImageObserver`

ABORT, ALLEBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

Constructors

Constructor and Description

`FusionToolUI()`
Creates new form FusionToolUI

Method Summary

Methods

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td>main(java.lang.String[] args)</td>
</tr>
</tbody>
</table>

This is the main function of this class.

Methods inherited from class `javax.swing.JFrame`

addImpl, createRootPane, frameInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getGraphics, getJMenuBar, getLayeredPane, getRootPane, getTransferHandler, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, repaint,
Methods inherited from class java.awt.Container

- add, add, add, add, addComponentListener, applyComponentOrientation, areFocusTraversableKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaxSize, getMinimumSize, getMousePosition, getPreferredSize, insets, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, processComponentEvent, remove, removeAll, removeContainerListener, removeComponentZOrder, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusDowncycle, validate, validateTree

Methods inherited from class java.awt.Component

- action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, clip, contains, contains, createImage, createImage, createImage, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBaseline, getBaselineResizeBehavior, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListener, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition,
getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getVisible, getFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundLayer, isBorder, isCursorLayer, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundLayer, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lastFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processM.btnDelete:**MouseMotionEvent**, processMouseWheelEvent, removeComponentListener, removeFocusListener, removeHierarchyListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocusInWindow, requestFocusInWindow, requestFocusInWindow, requestFocusInWindow, resize, resize, revalidate, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setMaximumSize, setName, setPreferredSize, show, size, toString, transferFocus, transferFocusBackward, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.awt.MenuContainer

getFont, postEvent

Constructor Detail

FusionToolUI

public FusionToolUI()

Creates new form FusionToolUI

Method Detail

main

public static void main(java.lang.String[] args)

This is the main function of this class.

Parameters:

args - the command line arguments
public class generateFiles
extends java.lang.Object

This class is responsible for generating and saving the result CSV file.

Constructor Summary

Constructors

Constructor and Description
generateFiles()
public void generateCsvFile(java.util.ArrayList<java.lang.String[]> List)

This method generates the result CSV file and displays a Save as dialog box for the user to select a folder to save the file in.

Parameters:

List - list of arrays that contains the records after merging the two datasets.
my.Fusiontool

**Class ManualFrame**

java.lang.Object
java.awt.Component
java.awt.Container
java.awt.Window
java.awt.Frame
javax.swing.JFrame
my.Fusiontool.ManualFrame

All Implemented Interfaces:

**public class ManualFrame**
**extends javax.swing.JFrame**

This class is for creating the window of the manual fusion method where the user has to select a common column in both datasets to use in matching across the datasets. It allows the user to select the common column from the datasets and directs the user to the next window.

**See Also:**
Serialized Form

**Nested Class Summary**

**Nested classes/interfaces inherited from class javax.swing.JFrame**
javax.swing.JFrame.AccessibleJFrame

**Nested classes/interfaces inherited from class java.awt.Frame**
java.awt.Frame.AccessibleAWTFrame

**Nested classes/interfaces inherited from class java.awt.Window**
java.awt.Window.AccessibleAWTWindow, java.awt.Window.Type

**Nested classes/interfaces inherited from class java.awt.Container**
java.awt.Container.AccessibleAWTContainer

**Nested classes/interfaces inherited from class java.awt.Component**
Field Summary

Fields inherited from class javax.swing.JFrame
accessibleContext, EXIT_ON_CLOSE, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

Constructors

Constructor and Description

ManualFrame()
Creates new form ManualFrame

This methods is the customise constructor.

Method Summary

Methods

Modifier and Type Method and Description
static void main(java.lang.String[] args)
The main function for the class

Methods inherited from class javax.swing.JFrame
getForeground, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLock, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getSize, getColor, getTreeLock, getValid, getX, getY, getFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMinimumSizeSet, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocusInWindow, requestFocusInWindow, requestFocusInWindow, resize, resize, revalidate, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusable, setForeground, setIgnoreRepaint, setLocale, setMaximumSize, setName, setPreferredSize, show, size, toString, transferFocus, transferFocusBackward, transferFocusCycle

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.awt.MenuContainer
setFont, postEvent

Constructor Detail

ManualFrame

public ManualFrame()
Creates new form ManualFrame

ManualFrame

This methods is the customise constructor. It has parameters that were passed from the previous window.

Parameters:
file1 - contains all the records of the first dataset.
file2 - contains all the records of the second dataset.
names1 - the fields name of the first dataset.
names2 - the fields name of the second dataset.
Method Detail

main

public static void main(java.lang.String[] args)
The main function for the class
Parameters:
  args -
## Class Check_existing_keys

```java
public class Check_existing_keys extends java.lang.Object
```

This class includes methods to check whether the key identifiers, namely GP, CCG, or Trust codes the user has selected from the first window, exist in the databases.

### Constructor Summary

#### Constructors

<table>
<thead>
<tr>
<th>Constructor and Description</th>
<th>Method Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is customize constructor.</td>
<td>Check_existing_keys()</td>
</tr>
</tbody>
</table>

### Method Summary

#### Methods

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>CCGDKKeyIdentified(int A, int B)</td>
</tr>
<tr>
<td>This method searches for the CCG and OP code in both datasets.</td>
<td></td>
</tr>
<tr>
<td>This method searches for the CCG code in both datasets.</td>
<td></td>
</tr>
<tr>
<td>This method searches for the GP code in both datasets.</td>
<td></td>
</tr>
<tr>
<td>This method searches for the Trust code in both datasets.</td>
<td></td>
</tr>
</tbody>
</table>

#### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

Check_existing_keys

public Check_exist_key()

This is customise constructor.

Method Detail

GPKeyIdentified


This method searches for the GP code in both datasets.

Parameters:
- FirstFile - contains all the records from the first dataset.
- SecondFile - contains all the records from the second dataset.

Returns:
- true if the key exist in the dataset, false otherwise.

CCGKeyIdentified


This method searches for the CCG code in both datasets.

Parameters:
- FirstFile - contains all the records from the first dataset.
- SecondFile - contains all the records from the second dataset.

Returns:
- true if the key exist in the dataset, false otherwise.

NHSKeyIdentified


This method searches for the NHS code in both datasets.

Parameters:
- FirstFile - contains all the records from the first dataset.
- SecondFile - contains all the records from the second dataset.

Returns:
- true if the key exist in the dataset, false otherwise.

CCGGPKeyIdentified

public boolean CCGGPKeyIdentified(int A, int B)
2/9/2014  

check_existing_keys

This method searches for the CCG and GP code in both datasets.

Parameters:

A - flag to check if the first file include GP and the second file include CCG
B - flag to check if the first file include GP and the second file include CCG

Returns:

true if the GP and CCG keys exist in the dataset, false otherwise.
**Class NextFrame**

java.lang.Object  
java.awt.Component  
java.awt.Container  
java.awt.Frame  
javax.swing.JFrame  
my.Fusionbol.NextFrame

All implemented interfaces:

```java
public class NextFrame  
extends javax.swing.JFrame
```

This class is for creating the window where the user selects the columns from both datasets to include in the generated CSV file. It is responsible for passing the selected columns to the fusion method where the two datasets are mapped and combined in one single dataset.

See Also:
Serialized Form

**Nested Class Summary**

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from class javax.swing.JFrame</th>
</tr>
</thead>
<tbody>
<tr>
<td>javax.swing.JFrame.AccessibleJFrame</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from class java.awt.Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.awt.Frame.AccessibleAWTFrame</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from class java.awt.Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.awt.Window.AccessibleAWTWindow, java.awt.Window.Type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from class java.awt.Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.awt.Container.AccessibleAWTContainer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from class java.awt.Component</th>
</tr>
</thead>
</table>
Field Summary

Fields inherited from class `javax.swing.JFrame`
- accessibleContext
- EXIT_ON_CLOSE
- rootPane
- rootPaneCheckingEnabled

Fields inherited from class `java.awt.Frame`
- CROSHER_CURSOR
- DEFAULT_CURSOR
- E_RESIZE_CURSOR
- HAND_CURSOR
- ICONIFIED,
- MAXIMIZED_BOTH
- MAXIMIZED_HORIZ
- MAXIMIZED_VERT
- MOVE_CURSOR
- N_RESIZE_CURSOR
- NE_RESIZE_CURSOR
- NORMAL
- NW_RESIZE_CURSOR
- S_RESIZE_CURSOR
- SE_RESIZE_CURSOR
- SM_RESIZE_CURSOR
- TEXT_CURSOR
- W_RESIZE_CURSOR
- WAIT_CURSOR

Fields inherited from class `java.awt.Component`
- BOTTOM_ALIGNMENT
- CENTER_ALIGNMENT
- LEFT_ALIGNMENT
- RIGHT_ALIGNMENT
- TOP_ALIGNMENT

Fields inherited from interface `javax.swing.WindowConstants`
- DISPOSE_ON_CLOSE
- DO_NOTHING_ON_CLOSE
- HIDE_ON_CLOSE

Fields inherited from interface `java.awt.image.ImageObserver`
- ABORT
- ALLBITS
- ERROR
- FRAMEBITS
- HEIGHT
- PROPERTIES
- SOMEDBITS
- WIDTH

Constructor Summary

Constructors:
- `NextFrame()` Creates new form NextFrame.
- `NextFrame(java.util.ArrayList<java.lang.String[]> file1,
java.util.ArrayList<java.lang.String[]> file2,
java.lang.String[] names1,
java.lang.String[] names2)`
  This methods is the customise constructor.

Method Summary

Methods:
- `static void findSelectedField()` This method retrieves the columns the user has selected and passes them to the suitable fusion algorithm based on the user selection.
- `static void main(java.lang.String[] args)` The main function of the class.
Methods inherited from class javax.swing.JFrame

- addImpl, createRootPane, frameInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getGraphics, getJMenuBar, getLayeredPane, getRootPane, getTransferHandler, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, repaint, setContentPane, setDefaultCloseOperation, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setIconImage, setJMenu, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, setTransferHandler, update

Methods inherited from class java.awt.Frame

- addNotify, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getJMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setBackground, setCursor, setExtendedState, setMaximizedBounds, setJMenuBar, setOpacity, setResizable, setShape, setState, setTitle, setUndecorated

Methods inherited from class java.awt.Window

- addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applicationResourceBundle, applicationResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBackground, getBufferStrategy, getFocusableWindowState, getFocusCycleRoot, getFocusOwner, getFocusTraversalKeys, getIconImages, getInputContext, getListeners, getLocation, getModalExclusionType, getMostRecentFocusOwner, getOpacity, getOwnerWindows, getOwner, getOwnerLessWindows, getShape, getToolkit, getType, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, hide, isActive, isAlwaysOnTop, isAlwaysOnTopSupported, isAutoRequestFocus, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isOpaque, isShowing, isValidateRoot, peekLayout, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, reshape, setAlwaysOnTop, setAutoRequestFocus, setBounds, setBounds, setC Salisbury, setFocusCycleRoot, setIconImages, setLocation, setLocationRelativeTo, getMinimumSize, getModalExclusionType, getSize, setSize, setPreferredSize, setSize, show, toBack, toFront

Methods inherited from class java.awt.Container

- add, add, add, add, add, ContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContents, getFocusCycleRoot, getFocusCycleRoot, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalid, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, List, locate, minimumSize, paintComponents, preferredSize, print, printComponents, processComponentEvent, remove, removeContainerListener, setComponentZOrder, setFocusCycleRoot, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

- action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createWindow, createWindow, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, fireComponentEvent, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyCh
Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.awt.MenuContainer

gGetFont, postEvent

Constructor Detail

NextFrame

public NextFrame()

Creates new form NextFrame

NextFrame

public NextFrame(java.util.ArrayList java.lang.String[] file1,
java.util.ArrayList java.lang.String[] file2,
java.lang.String[] names1,
java.lang.String[] names2)

This method is the customise constructor. It has parameters that were passed from the previous window.

Parameters:

file1 - contains all the records of the first dataset
file2 - contains all the records of the second dataset
Method Detail

FindSelectedFields

public static void FindSelectedFields()

This method retrieves the columns the user has selected and passes them to the suitable fusion algorithm based on the user selection.

main

public static void main(java.lang.String[] args)

The main function of the class

Parameters:

  args - the command line arguments
class ReadingFiles

This class deals with the CSV files.

Constructor Summary

Constructors

<table>
<thead>
<tr>
<th>Constructor and Description</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadingFiles()</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

Methods

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static java.util.ArrayList&lt;java.lang.String[]&gt;</td>
<td>readCSVFile(java.lang.String fileName)</td>
</tr>
</tbody>
</table>

This method reads a CSV file and returns it as a list of arrays so the other methods can easily deal with each line in the file as an array of data.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ReadingFiles:

public ReadingFiles()

Method Detail

readCSVFile

public static java.util.ArrayList<java.lang.String[]> readCSVFile(java.lang.String fileName)

This method reads a CSV file and returns it as a list of arrays so the other methods can easily deal with each line in the file as an array of data.

Parameters:

fileName - The File name

file:///home/servl_a/roc_mscsc13ya/NetBeansProjects/Fusiontool3/dist/javadoc/my/Fusiontool/ReadingFiles.html
Returns:
List of arrays contains all the records from the input file.