CITY UNIVERSITY LONDON

MSc Business Systems Analysis and Design

Project report

Academic Year: 2013/2014

Forever delayed: How the use of data modelling and visualisation techniques can provide rail commuters with more informed choices

Matthew Griffin

Project Supervisor: Prof. Jo Wood

Submitted: 26 September 2014

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

Signed: Matthew Griffin

Date: 24th September, 2014

Abstract

With the advent of real-time railway data being made available electronically to the general public, the opportunity exists to use this data to provide information regarding the actual performance of trains compared to their scheduled operations. By making this information available in visual format, users are offered the opportunity to make decisions based upon historic performance of trains. This research aimed to provide users with this information by collecting this data, transforming it into usable information, and then offering this information to users via both tabular-based and graph-based visualisations formats. This research then explores whether these visualisations can better inform users, and suggests further work that could have its basis on this research.

Acknowledgements

This thesis would not have been possible without the researcher's fiancée, Joanne Gibbins. Her love, compassion, and fortitude, made every word of this work possible.

Special thanks goes to Jo Wood, whose enthusiasm and encouragement was always appreciated, and whose eye for detail and creative acumen aided several design choices during the prototype phase. Thanks also to Jason Dykes and Neil Maiden, whose advice was beneficial during the selection and design phases of this thesis, and to Dr. Jon Eversham, who offered a critical eye and a vast collection of modelling and visualisation techniques.

The following individuals (knowingly or otherwise) provided data or source code that contributed to this thesis. Gratitude for their efforts goes to:

- Tom Cairns for the use of his API to access historic National Rail data,
- Carl Partridge for his work in combining station locations with their geographic data,
- Mike Bostock for his Marey chart example that was used as the building blocks for creating the graph visualisations, and
- Ignatius Teo (whose code is taken from http://www.sitepoint.com/data-structures-4/) for the Djikstra algorithm to determine train paths.

TABLE OF CONTENTS

1. Introduction

1.1. Background

As the population in major cities increases, the challenge for railway operators to move people around becomes increasingly problematic. In earlier times, it was simply easier to tear down and replace existing infrastructure with new systems that made better use of resources, and provided additional capacity for future growth. Today, the expense of replacing infrastructure, coupled with the inconvenience upon those relying upon the system, makes major overhauls difficult at best and impossible at worst. With passenger loads increasingly yearly, and the rate of infrastructure replacement not keeping pace with demand, railway operators need to find ways of engaging with passengers to provide them information regarding the likelihood of their service arriving on time.

Information currently provided by railway operators focuses primarily on their scheduled operation of services, with real-time information provided for active services based upon their current location within the system. This information is usually discarded from public access at the destination of the service. The majority of journey planners currently in existence focus on identifying routes that users can travel to their destination, without providing users any insight into whether the suggested route is actually viable given historic performance; it makes no sense to suggest a route that requires an interchange, when the interchange point is reached too late to make the connecting train.

For transport such as buses and planes, their paths are scheduled, but they are able to change their intended path in order to arrive at its destination with limited disruption. Trains, due to the nature of their design, cannot as easily deviate from their path in order to avoid obstacles, such as slower or disabled trains, blocked tracks, or bottlenecks within the system.

At time of writing, about one in every twenty people in the United Kingdom who commute from home to their place of work do so via the National Rail-owned railway system (Office of National Statistics, 2013). With an increasing population, it is expected more people will either choose or be forced to use trains to travel between locations. With infinite capacity and perfect reliability, this would not pose a problem for railway operators or passengers to flawlessly calculate their chance of arriving at their destination on time. However, *"only in an ideal world will all trains run exactly to scheduled time*" (Chymera and Goodman, 2012), and according to performance figures issued by the Office of Rail Regulation, one in every ten trains that ran in the United Kingdom during the 3rd quarter of 2013/2014 arrived more than five minutes after their scheduled arrival time (Office of Rail Regulation, 2014). While National Rail does make available performance statistics at a macro level,

this is of limited use to rail passengers; knowing that a train operator achieves 90% reliability does not mean that every train will achieve this level of reliability.

Data regarding the train movement of National Rail services has been made available under licence to the Association of Train Operating Companies (ATOC), who then use this data in their internal database known as Darwin. ATOC make available a limited set of data, consisting primarily schedules and live movement data, for developers to make use of for their applications (Raper, 2011).

Most existing projects that make use of this data do so with route planning in mind. The most popular of these is CityMapper, an iPhone application that allows users to calculate a route from any two locations covered by the application.

Currently, this is only publicly accessible when the data message is sent over the network; the data messages do not persist once the message is sent, thus being lost forever. Being able to capture this data at the time of broadcast and represent it in new ways can give rail users the opportunity to make more informed decisions regarding their choice of travel journey.

1.2. Research Question

This research will attempt to provide an answer to the following question:

How can data modelling and visualisation techniques be used to provide rail commuters and railway operators with more informed choices?

1.3. Aims, Objectives, and Motivations

The overall aim of this research was to determine whether the combined use of data modelling and visualisation techniques can effectively create representations of information that provide rail commuters with more intuitive information to plan their journey.

The objectives of this research were the following:

1. To design and implement a prototype system that produced two definitive outputs;
    a. A database that contains a full published list of scheduled trains run by specific train operating companies, and the actual performance of each train in the scheduled table, including system cancellations and alterations (where possible[1]), and

[1] Certain services and stations are not measured for performance, due to a lack of infrastructure.

    b. A set of visualisations that provides analysis of actual railway performance compared to the published schedule of train operations, with the aim of providing additional information to commuters about their travel needs.
2. To identify user stories and determine whether the visualisations are able to satisfy their needs.
3. To identify shortcomings in the new system, and suggest improvements that can be made

The ideal outcome for this research was to provide rail commuters with a more informed choice as to their travelling habits, and allow rail operators to more actively engage with commuters about their travelling choices.

This research focused solely on trains that are under the auspices of National Rail within the United Kingdom, operated by the following TOCs: Southeastern, Southern, First Capital Connect, South West Trains, and Eurostar UK (for which data is only available to the Channel Tunnel timing point in Folkestone).

Services that are currently operated by metropolitan transit providers such as Transport for London (the company responsible for the operation of the London Underground, Overground, and Docklands Light Rail services), as well as other public transport services including buses and trams, and private transportation such as taxis, are not included in this research. It is possible that further research could take into account the movement of other forms of transportation that are affected by issues relating to National Rail train movement.

The areas of schedule creation, service forecasting, ticket pricing, and user mobility, were not considered within the scope of this research. Any commercial usage or benefit of this research was not taken into consideration.

This research is structured in the following segments:

- The Literature and Technical Review section covers a review of existing literature regarding important facets of this research, as well as an overview of existing technical solutions.
- The Methods section covers the design, implementation, and expected outputs of the new system, as well as identifying user stories that will form the basis of the analysis of the visualisations.
- The Results section discusses the outcomes of the implementation of this new system, including

- The Discussion section covers the results as compared to the original objectives of this research.
- The Conclusion and Further Work section evaluates the project as a whole, and makes suggestions as to how this research can be extended in the future.

2. Literature and Technical Review

2.1. Overview

This section of the research covers a summary and analysis of existing literature regarding important areas of this work. This section also provides an overview and analysis of some technical work in the public domain, relating directly to the research question, that has influenced this research.

2.2. Visualisation of railway scheduling

Any research on visualising transportation must acknowledge the work of Etienne-Jules Marey, a French scientist whose graphical train schedule was first published in 1878 and has since been reproduced in many works, including on the cover of Tufte (2001), and as demonstrated by (Rougeux, 2011) and (Stanford Visualisation Group, 2010)). The major appeal of the Marey chart (also known as a stringline chart) is in its ability to plot multivariate data. As we are attempting to model event-time data for multiple outcomes of the same journey (or journeys) over a specified period of time, the Marey chart is appropriate to use for plotting railway performance data. This research attempted to extend the traditional Marey chart by combining the journey schedule (which would only replicate the original Marey chart) with each journey made for that schedule, as well as a line representing the average departure time (or arrival at the destination).

The work of Marey does appear to have influenced a section of the work of Kanai et al. (2011), which proposed a new algorithm for optimal delay management throughout the Japanese railway network. Kanai et al. explain the connection between movements of a pair of singular regular and express train through a short segment of railway using a Marey diagram in Figure 1, and then expand this out to multiple trains in Figure 2. Figure 10 of this work describes the behaviour model of a passenger making a decision as to their immediate route of transport. This research would augment this process by adding an extra step to this process, prior to the “Appear at Station” action of this figure. This research is independent of any passenger decision that is motivated by qualitative information not available to the system.

2.3. Railway reliability

As explained in the introduction, we cannot expect complete reliability from railway services. However, commuters may construct faulty mental models due to a lack of historic information, something that can have a “*powerful impact on … framing behaviour*” of their choices (Zhong, 2008). These mental models can be improved by providing information regarding their services that allow users to make more informed decisions.

2.3.1. Literature

The work of Robert Goverde on the performance of the Dutch railway network has focused on the punctuality of railways over many years. His work describes timetables as "*the backbone of scheduled railway systems*" (2005, pg. 4), and the responsibility for accuracy leads to the quality of *"effective railway capacity, traffic performance, quality of transport service, passenger satisfaction, train circulations, and schedules for railway personnel"*. The work in (2005) also touches upon the likelihood of trains to regain lost time; Goverde examines the impact of decisions made to reduce delays across the network. Goverde's research is generally highly quantitative, with data analysis combined with the application of algorithms to pursue the most optimal timetable for the Dutch system. Goverde admits that *"future effort must be directed towards usage of the empirical data in daily practice"*.

Chymera and Goodman (2012) describe the shift in scheduling of National Rail trains over time, from using train graphs to manually design schedules based upon empirical evidence and the reduction in conflicts between services, to the current system that now accepts data and calculates schedules from the entire United Kingdom railway system. They also describe in detail some of the practices utilised by train drivers and rail operators that affect the reliability of scheduling. The ability to analyse past information and apply it to future schedules would be of great benefit; this information must be provided in a format that would allow a much clearer overview of performance to those in charge of creating rail schedules.

2.3.2. Technical Review

This research concerns itself with providing rail commuters in the United Kingdom with information regarding the reliability of their chosen service. Currently, as described in the introduction, the only publicly available repositories of historic data are those written by developers whose software captures data messages sent from signalling points at the moment of transmission. These systems do provide similar data outputs relating to historic performance, but all of them crucially provide a point of difference unique to their system.

The first of these is **OpenTrainTimes**[2], a system designed in 2011 to capture live data and present current and historic train journeys in tabular format, and a route map of one particular journey. By providing the history of a particular journey, the system is able to provide information to a user on the entire lifetime of the journey, including actual performance compared to its scheduled movement. This is useful for passengers who need to confirm actual times for past journeys, possibly

[2] http://www.opentraintimes.com/

to claim compensation due to their train being late, but is of limited value when analysing expected performance due to the small sample size.

This system also provides a graphical visualisation showing live movement of trains between signalling points on a given line; while useful to railway controllers making decisions in real-time, this information is again of limited value to commuters making decisions based upon historic performance.

A system along similar lines is **Recent Train Times**[3], which has been in operation since late 2011. This system allows users to search for all trains between two selected stations, including such calculations as average arrival time at final destination, percentage of cancellations and arrivals within five minutes of schedule, and reporting best and worst performances over the time period requested. A user can access graphs that show daily arrival times for one or more trains at their destination, and also a modified histogram to show historic arrival within set intervals.

Realtime Trains[4] provides another example of providing primarily tabular-based railway reporting on historic performance. Realtime Trains also provides information on each station within the network, and all trains that pass through that station, whether or not the service has allowed passengers to alight; this information could be of use to railway operators, in determining optimal traffic flow and traffic routing through a particular station.

Another example of web-based performance system is **Raildar**[5], which uses National Rail data to provide data on several key metrics, such as public performance measure (PPM), cancellations, and total delay across the network. It is also possible to look at historic journeys in the same way as OpenTrainTimes with a similar tabular reporting format, but Raildar's one point of difference is that it provides a graph that displays the number of trains that have been delayed on route to the final destination of that train service.

2.4. Reliability of public transportation (excluding railways)

The analytical system developed by Ayhan et al., (2013) make use of data warehousing and real-time tracking techniques to make *"predictions based upon descriptive patterns of massive aviation data"*. This idea has similarities with the research this project has undertaken: the capture of near real-time data from moving transportation; the warehousing of large amounts of data; dealing with incorrect, incomplete, or incompatible data messages broadcast during the journey; design and

[3] http://www.recenttraintimes.co.uk/
[4] http://www.realtimetrains.co.uk/
[5] http://raildar.co.uk/

implementation of an architecture that will convert raw data into a usable visualisation format for analysis.

Due to the nature of intense data logging in the aviation industry, this level of detail cannot be matched when looking at railways. However, the two modes of transport share common data elements that allow a basic structure to be built

While the visualisation of real-time information appears uncommon in rail transportation literature, with the focus largely remaining on the scheduling of trains, the aviation industry provides a rich seam of literature that can be used as a starting block for information on the visual representation of data. Examples of this include Yao and Jiandong's research into a platform that can be used to predict delays for airports and airlines (Yao et al., 2009), using multi-dimensional scaling plots to flight durations and the effect of wind speed and visibility on traffic demand (Rehm et al., 2007, pg. 22–23), and determining the trajectory of aircraft using a bespoke visualisation tool (Santiago et al., 2009). the basic structure of rail data can be considered to be similar to those in aviation; primarily, a unique identifier for the route, its origin, its destination, expected and actual departure and arrival times, its current location, and any reasons for delays or cancellations.

2.5. The use of colour in visualisation

An important question is posed by Healey: *"how can we allow rapid and accurate identification of individual data elements through the use of colour?"* (1996, pg. 1). In the case of multiple train services, it quickly becomes clear that attempting to interpret graphical information, or even identify a single service, using a single colour is not optimal. The human eye will more readily distinguish patterns of information through the use of multiple colours, attaching relevant information to a colour palette. This does not mean using random colours, as this would not give information any additional meaning.

Lujin Wang et al. (Lujin Wang et al., 2008) then introduce an important element into visualisations, namely the ordering of semi-transparent elements in providing not only a *"pleasing image"*, but one that offers *"quick and accurate insight"* into the information being presented. This is a crucial consideration in this research – with multiple pieces of information being plotted in close proximity, it is vital to distinguish between individual trips; the idea of using opacity settings to allow patterns to be determined in the frequency of trains arriving at certain times was considered in this research.

It is also important to consider people with colour vision deficiency, or colour blindness, when delivering information that relies on colour to emphasise its message. Allport (1971) discovered that when combining multiple forms of information delivery, such as by using colour and shape together,

"*nearly perfect parallel encoding*" of information could be achieved. In this research, the combination of gradient indicating train movement, and colour indicating its proximity from its desired outcome, reduces the impact upon the approximately 8% of males and 0.5% of females who suffer from colour blindness (Simunovic, 2010)[6].

2.6. Tabular visualisation vs. Graph visualisation

A point that Friel et al., (2001) make in their work is that both tables and graphs can be considered to be forms of visualisation. However, it is important to note that the visualisations produced by data are only as useful as the knowledge of the reader. Presenting information without valid context can make even the most elaborate visualisation meaningless. Friel et al., also suggest that using line charts "*typically reflect functional relationships or time-series data*", which was appropriate given the expected output for this research.

Prasad and Ohja (2012) present a viewpoint that if tabular and graph visualisations are to be used, that steps should be taken to ensure "*that no explicit action* [is] *taken to maximize the salient features of any one particular modality*"; in other words, to present each form of visualisation as separately as possible. By doing so, the possibility of users being able to comprehend the information presented in one format due to having seen it presented in a different format is removed. Prasad and Ohja's experiment regarding modalities of information presentation suggest that users may find information faster using graphs, but accuracy of comprehension may be sacrificed in exchange for this speed. In terms of this research, people attempting to identify unreliable trains may come to a faulty understanding of what graph visualisations attempt to convey, if it is not clear what is being represented in those graphs; using tabular data provides much more definitive information, but it may take users longer to arrive at the same conclusion.

[6] Only relating to those with congenital colour blindness. Acquired colour blindness not included in this figure.

3. Methods

3.1. Overview

This research used a combination of quantitative and qualitative methods in order to produce its results. Quantitative data analysis (Oates, 2006, pg. 38) was used to calculate individual train performance over a given period of time. The original plan was adapted when it was discovered that different formats could be obtained for the same data, whilst an alternative source of performance data was discovered and used in place of the planned near real-time capture.

The primary focus of this research was to provide users with the quantitative data explained above. However, as the research also aimed to produce visualisations that allow users to make more informed decisions in regarding to selecting their journey, users' interpretation of these results needs to be taken into account. Information must be conveyed in a manner that users are able to make coherent decisions based upon this. By using qualitative analysis (Oates, 2006, pg. 38) in the form of task-based usability testing, feedback was generated regarding the appropriateness of the visualisations, and suggestions were made as to how to improve the final result.

By producing data in a variety of formats, it is possible for users to come to different conclusions, or to reach the same conclusion faster using one method over another due to personal strengths in either visual or numerical data processing.

3.2. Database Design

3.2.1. Data Sources and Capture

National Rail provides real-time information regarding train movement, signal information, delay and cancellation advice, and performance measurement. The Association of Train Operating Companies (ATOC) make available scheduling information for every major train (including passenger and freight) that operates in the mainland United Kingdom, as well as station locations and additional transit information between stations via other methods (walk, bus, Underground, etc.). Therefore, it is appropriate to use data generated by train movements and supplied by ATOC, through its National Rail Enquiries (NRE) data feed, for this research project.

The original proposal indicated that a program capable of live data capture would be used. This ultimately proved impractical due to the inability to handle the sheer volume of data produced by the capture, and so arrangements were made to use a REST API developed by a third party to gain

access to a database of historical railway performance. This differs from using the NRE feed, as the REST API allowed a “pull” of captured data, rather than relying on “push” data.

The original proposal also called for the access of National Rail schedules stored in a Common Interface File (CIF) format that contained identifying information on all trains. It was later discovered that these schedules are made available in a more user-friendly JSON format, which allowed for faster code development to handle the data.

An additional data source was used in the construction of the dataset, in the form of a file containing a conversion of physical station locations was included in order to provide distances ‘as the crow flies’ between stops. This allowed graph presentation to be displayed as a ratio between stops, allowing users to identify track segments that operate at different speeds. On occasion, the latitude and longitude co-ordinates giving for a particular stop proved to be erroneous, leading to nonsensical geographical representations; these co-ordinates were then checked for accuracy, and converted to their actual locations by using Google Maps to find the correct location of these locations.

3.2.2. Database Design and Decisions

National Rail scheduling includes any service that operates on its railway network. This includes information on passenger and freight trains, as well as stock movement throughout the network. It was decided to only include train services that were marked as ordinary passenger (coded as OO) or express (coded as XX) services for two reasons. Firstly, services that do not carry passengers, such as freight trains or railway stock moving between depots overnight, have been excluded due as these services cannot be used by passengers (though this data might be included in further work to determine the flow of train traffic as these services use the same infrastructure as regular passenger trains). Secondly, irregularly scheduled or emergency services, such as those that run on Bank Holidays, have been excluded due either to a lack of data, or to the fact that unscheduled trains cannot have historic performance information to compare.

It was also decided that with around 20,000 services operating on any given weekday in the United Kingdom, the focus of data would be on four major railway operators located in the southeast of the United Kingdom; Southeastern, South West Trains, First Capital Connect, and Southern trains. Data was included for Eurostar passenger services from St Pancras International to Ebbsfleet, the last station on the United Kingdom leg of Eurostar services, as these service add an extra level of complexity; where normal passengers trains can be boarded within 30 seconds of departure, Eurostar services require check-in no later than 30 minutes prior to departure.

During the programming phase, it was discovered that using the National Rail provided geographic locations could not be used due to the additional calculations required to generate latitude and longitude co-ordinates for the graph visualisations. This was discovered at an early enough stage in the project that an alternative dataset could be sourced and implemented, as discussed in section 3.2.1.

The database structure was decided to be split into five separate tables, in order to minimise repetitive data while increasing query performance. These were:

- A table for holding information regarding the schedule
- A table for holding information regarding each stop on the schedule
- A table for holding information regarding each railway station
- A table for holding information regarding historic performance, where data for the timing point at each station was captured
- A table for holding information regarding fixed-point links between two stations (see section 6.2)

Once the data files were acquired and analysed, an entity relationship diagram (ERD) was designed to provide the overall physical structure of the database, and to determine the primary and foreign keys for the specified tables. A relational model was generated from this ERD, in order to provide full documentation of the design specification for the purposes of traceability (Oates, 2006, pg. 113) and to allow this research to be replicated if necessary.

This research uncovered several issues that were not readily apparent at the commencement of this project:

- A train make stop multiple times at one station on any given journey, due to the fact that some trains operate in a loop (that is, their origin is also their destination). Thus, using the station identifier to determine the path of the train was not satisfactory, and a counter was introduced to determine the stop that was required for returning the correct information.
- Trains operating in a loop format (see above) may run in different directions at similar times of the day. This means that one service may run clockwise and the next service may run counter-clockwise. In an attempt to prevent users being shown irrelevant data, a selection was added such that users could select a station that is on their desired route. Note that users should be encouraged to select a station close to their destination.

- A scheduled train may have multiple schedules, due to changes for operational reasons. Thus, there may be more than one schedule train per identifier. Each set of stops therefore was required to contain the start and end dates of each valid schedule.
- Trains may stop at all stations along a route, or they may not pass through stations without stopping. This needs to be taken into consideration when displaying graph information.

3.3. Programming Languages and Tools

The schedule parser, data extractor from the REST API, all insertion and retrieval into the system database, and the tabular visualisation generator, were all written in the PHP programming language. The database was created using a MySQL database server. These tools were chosen for two primary reasons; firstly, the researcher was most comfortable using these tools for web scripting rather than using a Windows solution, and secondly, these visualisations could then be presented in a web browser, something that nearly all computer users have access to.

The graph visualisations were created using a combination of JavaScript and the D3 programming languages (see section 3.4 for further details).

Programming took place using the NetBeans IDE, as this offered a simple form of version control that was used whenever it became necessary to rollback to an earlier iteration of code. The code and database were accessed using an installation of WAMP, a popular package for internet development and deployment.

3.4. Visualisations

It was decided that both tabular and graph visualisations should be produced as outputs for this research, as it was felt that some of the generated information could not be shown successfully using graph visualisations.

The key pieces of information that was displayed consistently across both sets of visualisations for commuters were originating station, destination station, scheduled time of departure from origination, scheduled time of arrival at destination, and average time across all journeys. It was decided that this was the absolute minimum that users would accept when choosing a train service.

Initially, it was expected that the Processing programming language would be used to produce all visualisations used in this research. After additional research into available technologies and clarification of visualisation categories, it was decided to use PHP and MySQL to generate the data

for all visualisations and to produce visualisations in tabular format, and JavaScript and the D3 add-on developed by Mike Bostock to create the graph visualisations.

3.4.1. Graph output formats

The decision was taken to base the graph visualisations upon the Marey charts described in the Literature Review, as this is a commonly accepted visualisation template for displaying railway movement.

This research makes subtle changes to the original chart layout, taking advantage of the interactive nature of the new system. In the original Marey diagram, train movement is only displayed from their origin to their destination, meaning that passengers have to search for their station. Users will be able to select any station along the route that their service calls upon, not just the intending originating station for the journey.

In addition, the length of time a train spends stationary at a station is displayed as a horizontal line on the original Marey chart, while this research does not take into account time spent at a station, as passengers are more likely to be concerned with the arrival time at their destination than with the time spent waiting at any intermediate station.

It was decided to display a scheduled train using a solid black line, and the average performance at each stop using a solid blue line use a palette range of four colours when producing the results of historic performance within the graph visualisations, ranging from green for trains arriving less than one minute late, to purple for trains running significantly late. While this is in conflict with Healey's research that users were able to distinguish information with more accuracy when the number of colours used was no more than 5 (1996), it was decided that users would easily be able to distinguish between the use of two distinct colours to display singular pieces of information (the schedule, and the average of all historic performances) and the use of a four colour range (green, yellow, red, and purple) for each tracked journey.

A paper prototype was sketched out to envisage how the final graph visualisation would appear to users, as it was believed that this would "encourage interactive exploration of different implementation possibilities" (Kazman and Carriere, 1996, pg. 1). A series of prototype visualisations were built in order to ensure that the correct data was being captured and displayed. Initially, the prototype presented one schedule and one actual train journey (see Figure 1), then one schedule and multiple journeys (see Figure 2), and finally multiple schedules and multiple journeys (see Figure 3).

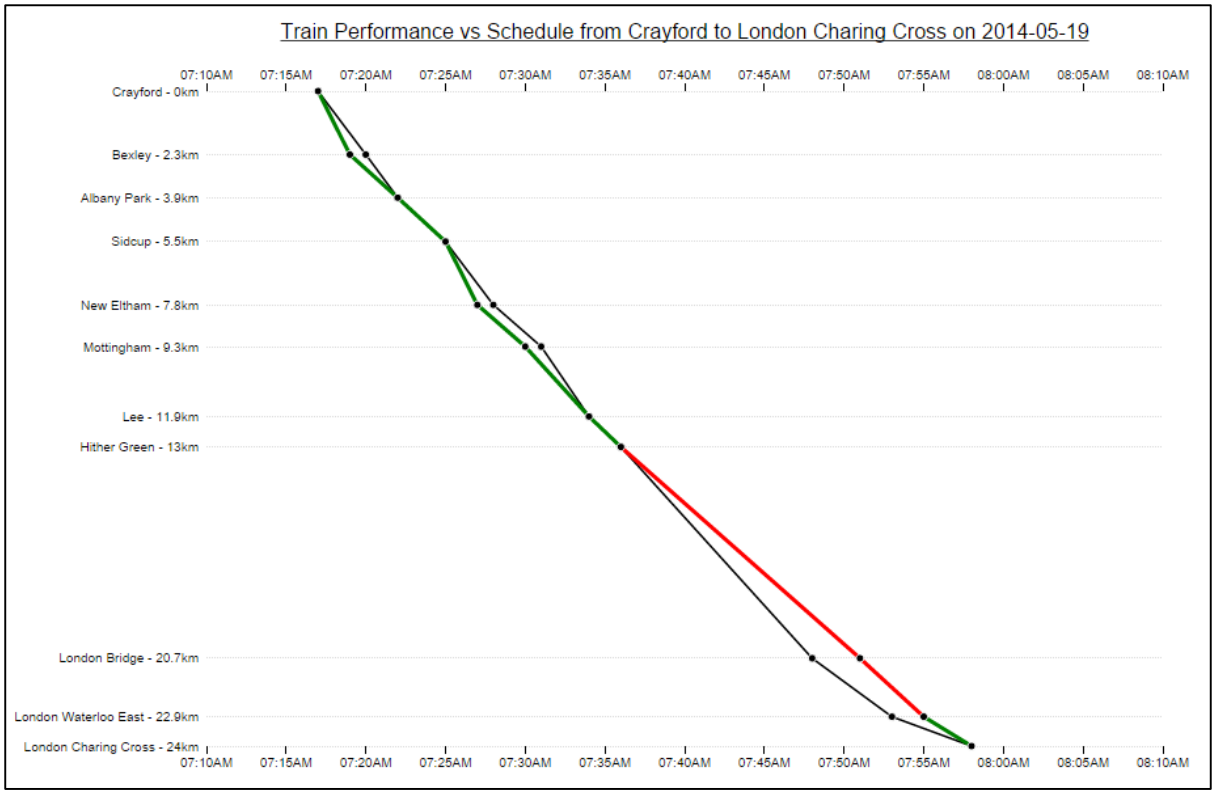


Figure 1. An example graph visualisation for a single service.

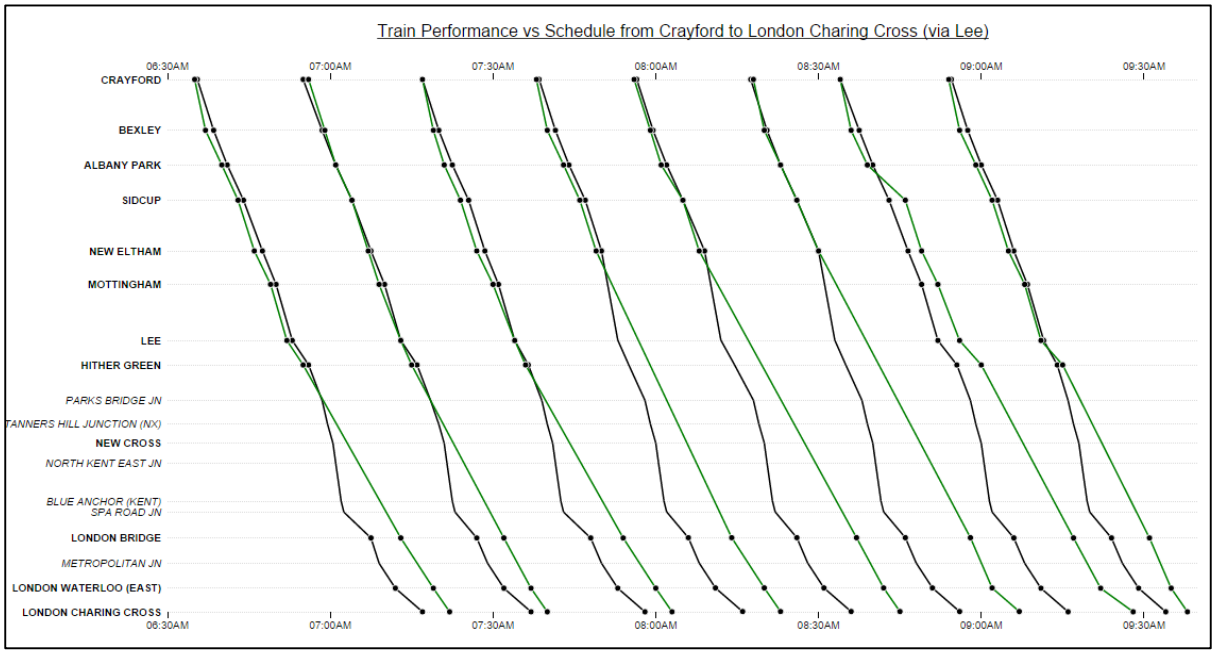


Figure 2.An example graph visualisation for multiple services.

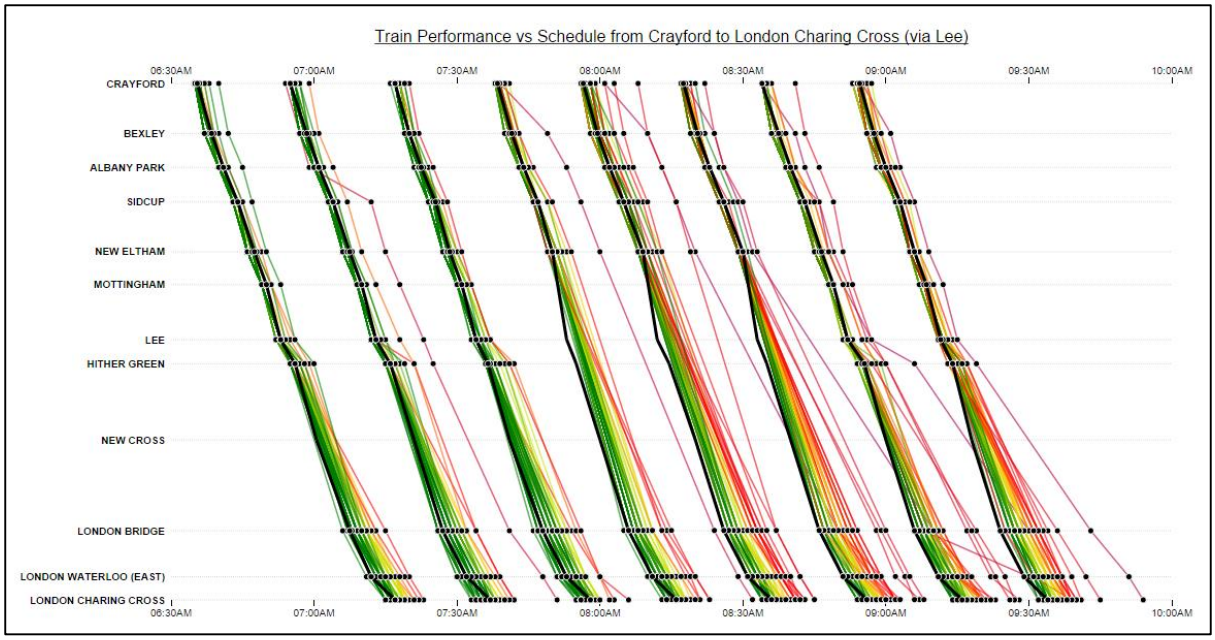


Figure 3. An example graph visualisation for multiple services, including multiple journeys.

3.4.2. Tabular output formats

Tabular visualisations are commonplace throughout the world of business. Most people are familiar with the grid-based outputs on software such as Microsoft Excel. This research produced tabular visualisations that were able to display information on each train journey from the user selected origin to destination stations; scheduled departure and arrival times, number of services ran, number of services cancelled, number of trains that arrived at their destination on or before scheduled arrival time, percentage of trains on time, percentage of trains that arrived within 5 minutes of schedule, average arrival time, and the time that 90% of trains arrived by. Arrival times at destination for each journey were then displayed to the right of this statistical information.

One of the primary drawbacks of the Marey chart is in its inability to chart more than one route at a time. From the perspective of a railway operator running multiple lines, this is not an optimal outcome. However, using data represented in a tabular output can provide an operator with the thoroughfare activity information at one particular station. This research provides a tabular visualisation that allows users to see all trains scheduled to transit through a selection station for a selected time period.

3.5. Programming Methodology

An iterative and incremental development methodology was used in building the prototype (Larman and Basili, 2003). This was considered to be the most appropriate method to use in this research, as

each section of the prototype needed to be built before being able to progressing to the next section. With time being a crucial factor in the development process, being able to produce partial output early in the life cycle was important in order to demonstrate progress. In some cases, modules that had been marked as completed in earlier phases had to be reworked when it was discovered that data was being incorrectly manipulated by later modules.

Using an incremental methodology meant that testing could take place as development occurred, reducing the likelihood of errors creeping in later phases, while ensuring that any changes necessary could be identified and implemented without delay.

3.6. User story scenarios

It was decided that there were three key areas that railway commuters and employers would derive the greatest benefit from by using this system.

The first was the reliability of a given service over time for the purposes of regular commuting. Regular commuters, who were dissatisfied with the performance of their current service, or those who were changing services due to external factors such as moving to a new area or changing work shifts, would receive the most benefit from being able to identify the probability of their train arriving at its destination on time.

The second area was reliability of a given service to arrive at its destination for the purposes of a one-off journey, which would be of benefit to users who use trains rarely, or tourists visiting a new city, These rail users would need to feel confident that the train that they have selected will arrive at a time that they expect, and this prototype would be required to show them recent performance of the train they have selected, in a format that they could understand with as short a learning curve as possible.

Thirdly, it was decided that railway operators would be interested in the frequency of all services at any given station over a specific period of time. This could allow better scheduling to take place if bottlenecks are identified, as well as being used to better plan staffing levels during busy periods.

3.7. Testing and Tasks

To determine how usable these visualisations would be to users, as well as attempting to identify shortcomings in the visualisations, task-based usability testing was then performed (Rubin and Chisnell, 2008) with a selected group of five participants whose opinions were sought on the visualisations that had been generated. Five were selected as it was decided that the majority of obvious shortcomings would be discovered by using this many representative users (Nielsen, 2000).

These users were selected from contacts of the researcher, as all five had used rail transport for different reasons.

Participants were asked to complete three separate tasks;

- To determine the average time for a given route, both using tabular and graph visualisation,
- To determine the time that passengers could expect their train to arrive at their destination with 90% confidence, again using both visualisation methods, and
- To determine the number of trains that passed through a given station in a set time period.

Participants were then given a survey form which asked them to rate how they found elements of the visualisations. Participants were also asked to make any free comments that they had regarding either of the visualisations.

3.8. Recruitment of participants

Potential participants in the study were contacted by the research in order to determine suitability for usability testing. It was decided that a baseline level of comprehension was required to participate; namely, that users had to have been based in the United Kingdom to allow them to attend testing, and that they had to be familiar with National Rail trains.

Participants were required to sign a consent form, which was adapted from a suggested template and presented for approval during the proposal phase of this project (please see Appendix A).

3.9. Usability testing outcome

Once a stable prototype had been designed and implemented, five users were selected to conduct task-based usability testing on the prototype, with the purpose of detecting errors or omissions in the visualisations, as well as providing feedback on the visualisations themselves.

Nearly all users were able to distinguish that each individual line represented one journey. Several also understood that the black and the blue line represented aggregated data (though it was pointed out that there was no way of identifying without highlighting the line what these represented, however, and this was identified as a weakness in the prototype that was rectified). Testers reported that while the individual lines themselves revealed little about their journey, being able to identify the range of times that trains arrived at their destination meant that they could easily determine whether the average was being skewed by a handful of extremely late arriving services.

Several testers reported that the information regarding individual performance was unnecessary on the tabular visualisation; it was suggested that providing an aggregate summary would be of more use to system users, and that individual performance would be of interest only if the user had a vested interest in that particular service (i.e., for the purposes of gaining compensation for a delayed train). All users recommended that a total be made available on the station performance table, as it would be of more benefit to know how many services were included on this visualisation.

There were also issues regarding the labelling of information (such as defining what constituted a "service" compared to what constituted a "train"), as well as minor modifications that were made before the final prototype was completed.

4. Results

4.1. Data generation

This research captured 30,560 scheduled rail journeys.

This research captured data for 664,286 possible journeys over a time period of 19th May 2014 to 24th August 2014. Additional data captured before 19th May 2014 was not used in this project due to the fact that new schedules were implemented on that date.

These captured journeys were broken down by the following:

WEEK COMMENCING	Mon	Tues	Wed	Thurs	Fri	Sat	Sun	WEEK TOTAL
19/05/2014	7,340	7,356	7,359	7,360	7,366	6,574	4,690	48,045
26/05/2014	10,763*	7,388	7,365	7,367	7,371	6,992	4,583	51,829
02/06/2014	7,347	7,363	7,365	7,367	7,371	6,735	4,459	48,007
09/06/2014	7,341	7,354	7,356	7,358	7,360	6,609	4,528	47,906
16/06/2014	7,352	7,364	7,366	7,368	7,365	6,571	4,404	47,790
23/06/2014	7,344	7,355	7,357	7,359	7,364	6,572	4,464	47,815
30/06/2014	7,341	7,357	7,359	7,361	7,366	6,694	4,356	47,834
07/07/2014	7,337	7,355	7,357	7,359	7,363	6,571	4,254	47,596
14/07/2014	7,335	7,353	7,355	7,357	7,362	6,571	4,254	47,587
21/07/2014	7,335	7,353	7,355	7,357	7,362	6,575	4,254	47,591
28/07/2014	7,333	7,353	7,353	7,355	7,360	6,575	4,253	47,582
04/08/2014	7,337	7,353	7,353	7,355	7,360	6,575	3,972	47,305
11/08/2014	6,722	6,732	6,733	6,734	6,739	5,962	3,989	43,611
18/08/2014	6,723	6,733	6,733	6,735	6,739	6,084	4,041	43,788
DAY TOTAL	104,950	101,769	101,766	101,792	101,848	91,660	60,501	664,286

Table 4.1. Number of journeys captured per day, displayed by week.
** Higher than usual number due to replacement train services run on Bank Holiday on 26tH May, 2014.*

4.2. Visualisation generation

Visualisations are provided in two formats, tabular and graph, which users can select from on the selection screen (Figure 4.2); users can switch to the graph visualisation from the tabular visualisation on the train journey report visualisation (Figure 4.3). These formats and the results they produced are examined separately below.

4.2.1. Tabular visualisations

There are three tabular visualisations made available to users, depending on the selections that they make. The first visualisation provides users with a list of trains that travel from their selected origin

through to their selection destination, passing through a selected station, and commencing between the times that they have selected. A user may select any range of 24 hours, but should be aware that trains running after midnight will be reported in the next day's summary.

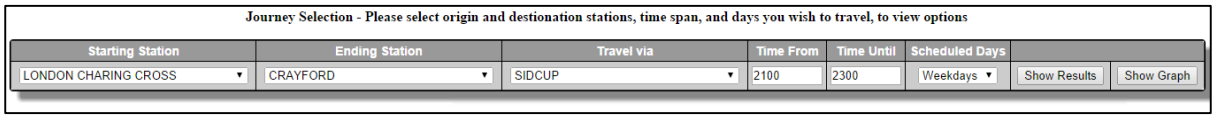


Figure 4.1. Selection screen for users; currently selecting trains from London Charing Cross to Crayford via Sidcup.

In Figure 4.1, the user has selected to travel from London Charing Cross, travelling to Crayford in Kent, via Sidcup, between 2100 (9:00pm) and 2300 (11:00pm) in the evening. The system returns four possible journeys that could be made in that time, as well as aggregated information regarding each journey (see Figure 4.2). What this shows is that while only between 20% and 35% of journeys arrive on time, less one in ten trains will arrive more than five minutes after scheduled.

Daily train performance - trains departing London Charing Cross between 2100 and 2300 and arriving at Crayford

Train ID	Signalling ID	Days Run	Departure (London Charing Cross)	Arrival (Crayford)	Average Arrival Time	90% Arrival Time	Number of Trains Run	Number of Trains Not Run	On Time	% On Time	% within 5m
W09543	2N74	Mon-Fri	21:22	22:02	22:04	22:07	59	1	14	23.7%	91.5%
W09547	2N76	Mon-Fri	21:52	22:32	22:34	22:37	59	1	12	20.3%	91.5%
W09552	2N78	Mon-Fri	22:22	23:02	23:03	23:07	59	1	18	30.5%	91.5%
W09556	2N80	Mon-Fri	22:52	23:32	23:33	23:37	59	1	20	33.9%	91.5%

Figure 4.2. Tabular visualisation of data returned from selections made in Figure 4.1.

The second visualisation returns results of all journeys scheduled for a particular service; this report is linked from the main list of trains, and must be selected by the user in order to get access to this secondary level of information. This report is used for determining the result of each journey, whether or not it completed the journey. Cancellations are noted, as well as the reason why the journey was cancelled, if that data is made available.

Daily information for the train from Gravesend (departure 0932) to London Charing Cross (arrival 1037) - all movement at London Charing Cross

Train ID	Signalling ID	Train Operator	Days Run	Location Type	Date Run	Stop Number	TIPLOC Code	WTT Arrival	WTT Departure	Public Arrival	Public Departure	Actual Arrival	Actual Departure	Minutes Late	Reason for Cancellation
W07700	2D24	SE	Sun	LT	18/05/2014	29	London Charing Cross	10:37		10:37		10:38			
W07698	2D24	SE	Mon-Fri	LT	19/05/2014	27	London Charing Cross	10:01		10:01		10:04			
W07698	2D24	SE	Mon-Fri	LT	20/05/2014	27	London Charing Cross	10:01		10:01		10:03			
W07698	2D24	SE	Mon-Fri	LT	21/05/2014	27	London Charing Cross	10:01		10:01		10:05			
W07698	2D24	SE	Mon-Fri	LT	22/05/2014	27	London Charing Cross	10:01		10:01		10:06			
W07698	2D24	SE	Mon-Fri	LT	23/05/2014	27	London Charing Cross	10:01		10:01		10:03			
W07705	2D24	SE	Sat	LT	24/05/2014	27	London Charing Cross	10:00		10:00		10:03			
W07700	2D24	SE	Sun	LT	25/05/2014	29	London Charing Cross	10:37		10:37		10:37			
W07698	2D24	SE	Mon-Fri	LT	26/05/2014	27	London Charing Cross	10:01		10:01					The planned cancellation (was not planned to operate)
W07698	2D24	SE	Mon-Fri	LT	27/05/2014	27	London Charing Cross	10:01		10:01		10:03			
W07698	2D24	SE	Mon-Fri	LT	28/05/2014	27	London Charing Cross	10:01		10:01		10:06			

Figure 4.3. Tabular visualisation showing information on all journeys for the scheduled 0932 (9:32am) service from Gravesend to London Charing Cross (extract shown here). Note planned cancellation on 26/05/2014.

The third visualisation is a report showing all trains that are scheduled to pass through a selected station, as well as providing information on the average arrival and departure time for each train. In

Figure 4.4, the user is provided with a list of 116 trains travelling through Crayford between 0700 (7:00am) and 2000 (8:00pm), with the average arrival and departure times shown clearly. In this case, we can see that trains travelling through Crayford tend towards an extremely high probability of arriving within five minutes of their scheduled arrival time. This visualisation can also identify trains that have a high probability of unreliability through busy stations. Figure 4.5 shows traffic flow through London Blackfriars between 1000 (10:00am) and 1100 (11:00am), and trains can be identified from this visualisation as almost never adhering to their scheduled timing.

Daily train performance - trains calling at Crayford between 0700 and 2000 (weekday services only) - 116 services

Train ID	Signalling ID	Train Operator	Days Run	Origin	Destination	Arrival Time	Departure Time	Average Arrival Time	Average Departure Time	Trains Run	On Time	% On Time	% within 5m
W09395	2N08	SE	Mon-Fri	LONDON CHARING CROSS	GRAVESEND	07:07	07:07	07:06:37	07:07:09	59	46	78%	98.3%
W07600	2D09	SE	Mon-Fri	CRAYFORD	LONDON CANNON STREET		07:10		07:10:05	59	52	88.1%	98.3%
W07628	2D14	SE	Mon-Fri	GILLINGHAM (KENT)	LONDON CHARING CROSS	07:17	07:17	07:16:19	07:17:20	57	39	68.4%	98.2%
W07613	2D11	SE	Mon-Fri	CRAYFORD	LONDON CANNON STREET		07:30		07:30:04	55	50	90.9%	98.2%
W09397	2N10	SE	Mon-Fri	LONDON CHARING CROSS	GRAVESEND	07:37	07:37	07:36:09	07:36:57	59	52	88.1%	98.3%
W05911	1D50	SE	Mon-Fri	GRAVESEND	LONDON CHARING CROSS	07:38	07:38	07:37:17	07:38:35	57	26	45.6%	98.2%
W09079	2M10	SE	Mon-Fri	LONDON CHARING CROSS	CRAYFORD	07:48		07:46:53		59	51	86.4%	96.6%
W07624	2D13	SE	Mon-Fri	CRAYFORD	LONDON CANNON STREET		07:50		07:50:14	58	52	89.7%	100%
W09401	2N11	SE	Mon-Fri	LONDON CANNON STREET	SLADE GREEN	07:50	07:50	07:49:52	07:50:37	58	36	62.1%	98.3%
W05912	1D52	SE	Mon-Fri	GILLINGHAM (KENT)	LONDON CHARING CROSS	07:56	07:56	07:56:25	07:57:21	59	26	44.1%	91.5%
W09402	2N12	SE	Mon-Fri	LONDON CHARING CROSS	GRAVESEND	08:04	08:05	08:03:54	08:05:20	59	48	81.4%	96.6%

Figure 4.4. Tabular visualisation showing information all trains stopping at Crayford between 0700 (7:00am) and 2000 (8:00pm) (extract shown here).

Daily train performance - trains calling at London Blackfriars between 1000 and 1100 (weekday services only) - 24 services

Train ID	Signalling ID	Train Operator	Days Run	Origin	Destination	Arrival Time	Departure Time	Average Arrival Time	Average Departure Time	Trains Run	On Time	% On Time	% within 5m
G71373	2O22	FC	Mon-Fri	SUTTON (SURREY)	LUTON	09:59	10:00	10:01:45	10:02:46	57	25	43.9%	78.9%
G71405	2O33	FC	Mon-Fri	ST ALBANS CITY	SUTTON (SURREY)	10:00	10:00	10:01:06	10:02:47	59	47	79.7%	83.1%
G71571	2T25	FC	Mon-Fri	BEDFORD	BRIGHTON	10:05	10:05	10:05:42	10:06:37	58	43	74.1%	91.4%
G71143	1T10	FC	Mon-Fri	BRIGHTON	BEDFORD	10:08	10:08	10:13:23	10:14:28	55	0	0%	34.5%
G71318	2E25	FC	Mon-Fri	KENTISH TOWN	SEVENOAKS	10:12	10:12	10:10:53	10:12:37	58	52	89.7%	94.8%
G71770	2V24	FC	Mon-Fri	SUTTON (SURREY)	ST ALBANS CITY	10:12	10:14	10:15:00	10:16:15	58	24	41.4%	89.7%
G71802	2V35	FC	Mon-Fri	LUTON	SUTTON (SURREY)	10:16	10:16	10:15:56	10:17:31	59	48	81.4%	91.5%
W08237	2E73	SE	Mon-Fri	SEVENOAKS	KENTISH TOWN	10:18	10:18	10:19:00	10:20:11	55	9	16.4%	85.5%
G71167	1T19	FC	Mon-Fri	BEDFORD	BRIGHTON	10:20	10:20	10:22:34	10:23:30	58	19	32.8%	77.6%
G71563	2T22	FC	Mon-Fri	BRIGHTON	BEDFORD	10:23	10:24	10:29:39	10:30:44	59	1	1.7%	37.3%
G71124	1J90	FC	Mon-Fri	BEDFORD	ELEPHANT & CASTLE	10:26	10:27	10:28:10	10:29:44	58	39	67.2%	79.3%
G71385	2O26	FC	Mon-Fri	SUTTON (SURREY)	LUTON	10:27	10:30	10:29:49	10:31:24	57	29	50.9%	89.5%

Figure 4.5. Tabular visualisation showing information all trains stopping at London Blackfriars between 1000 (10:00am) and 1100 (11:00am) (extract shown here). Note the 10:08 and 10:23 trains from Brighton – only one train in 120 arrived on time!

4.2.2. Graph visualisations

Users are able to hover over lines to show the actual performance over that particular journey; the colour displayed indicates quickly to the user the overall punctuality of that journey. In Figure 4.5, the green line in the first set of train journeys indicates that despite the fact that it departed late from its origin station (Crayford), the train was able to arrive slightly ahead of its scheduled time. Compare this to Figure 4.6, where a delay between Albany Park and Sidcup has resulted in the train arriving very late at its destination (indicated by the red line) in the second set of train journeys.

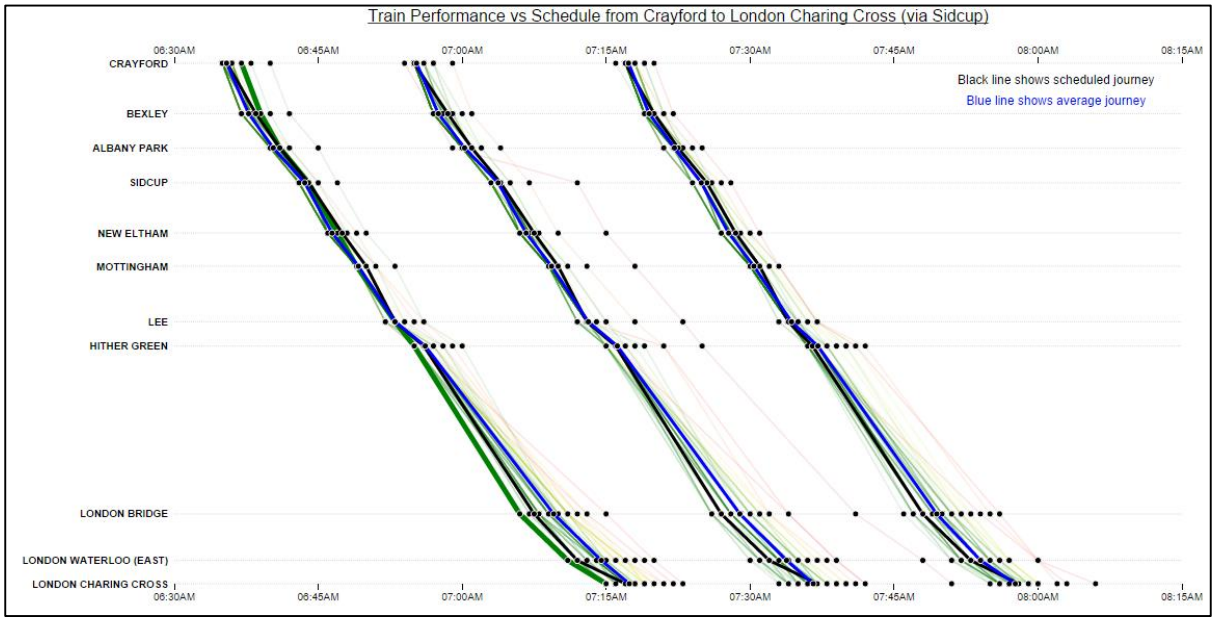


Figure 4.6. This is an example of a highlighted journey that arrived slightly ahead of schedule (the green line in the first set of train journeys). This shows weekday trains from Crayford to London Charing Cross between 6:30am and 7:30am.

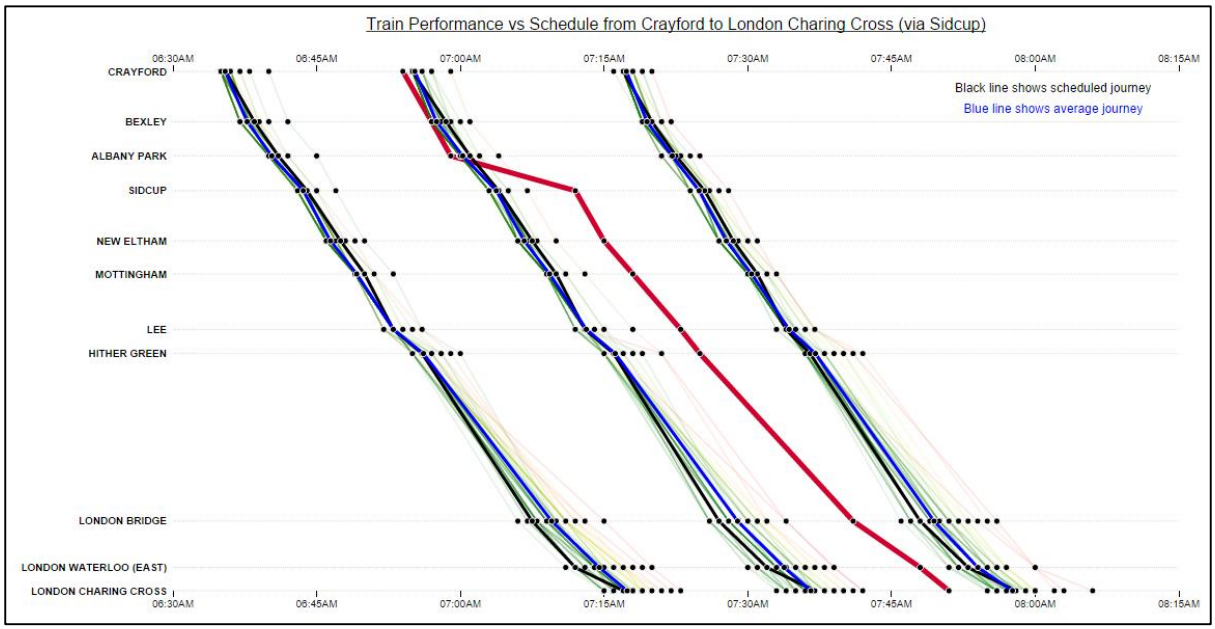


Figure 4.7. This is an example of a highlighted journey that arrived over 10 minutes late at its destination (the red line in the second set of train journeys).

Users also have the ability to highlight any station that a train stopped at on a particular journey, with a pop-up box showing information regarding that stop on the journey (refer to Figure 4.7 for an example); station name, date this journey took place, train identifier, and scheduled and actual departure times, are returned on screen to provide precise information to the user.

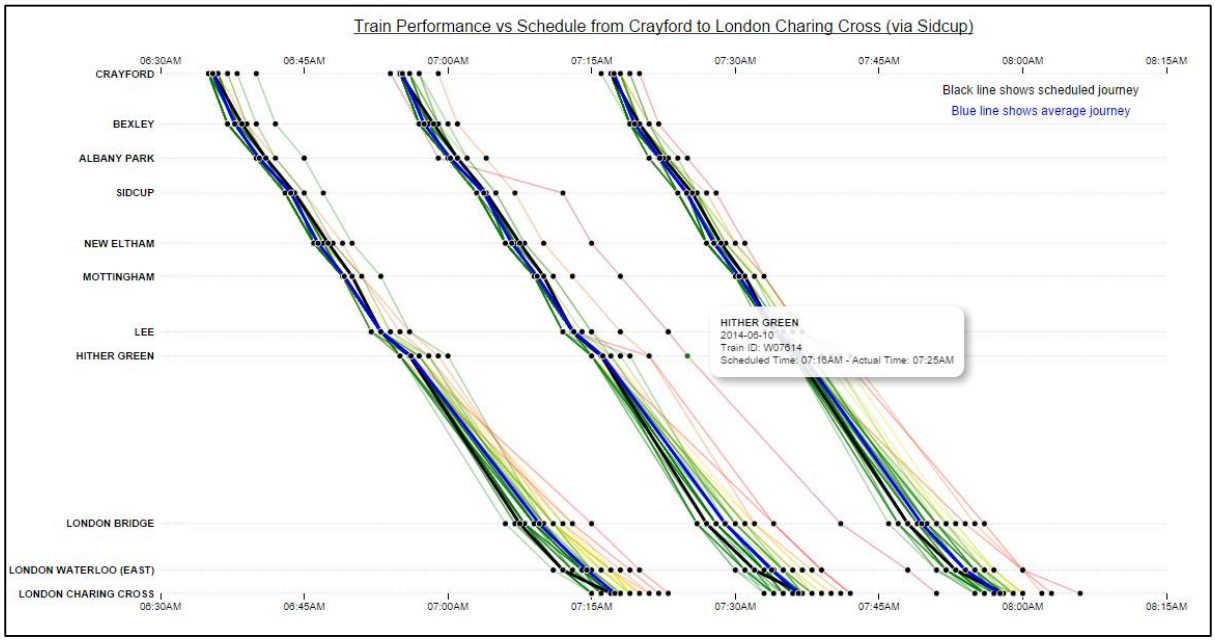


Figure 4.8. An example of station information for a selected journey, using a pop-up box that shows users additional information about the journey. The pop-up box displays the fact that this journey was 6 minutes late at Hither Green.

This prototype can be used to show trains over a much larger period of time, in order to identify trends in reliability. In Figure 4.8, peak hour trains are selected travelling from Brentford to London Waterloo. These services originate from London Waterloo and travel in a loop around Surrey, and Brentford is an intermediate stop. This graph shows that as the peak hour rush progresses, the range of journey start times increases, and the range of arrival journey times is also much wider than earlier trains. Users can also identify trains have been dramatically delayed between two stations, possibly due to train breakdowns or signalling issues on the network. One interesting trend is that alternate trains appear to suffer from more delays. In Figure 4.8, every second train appears to have a wider starting time range than every other train.

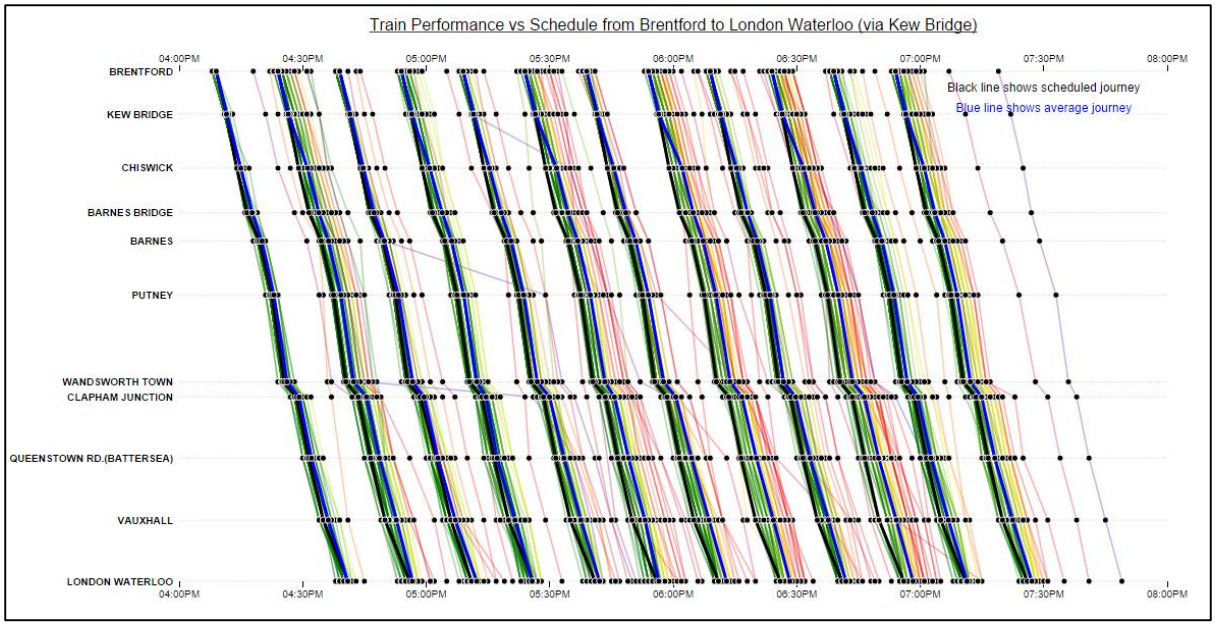


Figure 4.9. All weekday trains travelling from Brentford to London Waterloo between 1600 (4:00pm) and 1930 (7:30pm).

5. Discussion

5.1. Project deliverables

This research has been able to accept two data sets, one containing scheduled information about train services, and one that contains timing data about each stop on selected train journeys. By identifying common elements to both sets, namely the train's schedule identifier and station identifier, train performance against scheduled information can be measured using captured timing data.

This research has also been able to produce two distinct visualisation formats that show the performance of journeys over a given amount of time; in the case of this research, about three months of data was captured and used for this project. The first visualisation format was tabular visualisation, where aggregated information was presented numerically in a table alongside service information. The second visualisation format was graph visualisation, where aggregated information was presented alongside observed journey information, with each station on the vertical axis and the time on the horizontal axis.

The researcher notes that this data was captured during the summer, when weather impact is expected to be less than during times of turbulent weather; it is expected that train reliability would decrease during months where weather is less advantageous.

5.2. Case studies

There were three main areas that were identified as being relevant to commuters and railway staff: the reliability of their regular service; the reliability of a one-off journey; the throughput of a particular station during a given period of time.

5.2.1. Regular commuters

Railway users who regularly commute between two locations would be most interested in two pieces of information – their expected arrival time at their destination, and their required departure time from their origin station. Currently, commuters must rely on the published schedule, plus their own empirical evidence regarding their journey, to decide whether they will continue to use their regular service, or change to an earlier or later service.

By offering information in both tabular and graphic formats, users have the opportunity to make their decisions based upon the format of data that allows them to feel most comfortable. All testers felt at least somewhat comfortable about using either visualisation technique, although tabular

information was described as being slightly more useful than the graph information. This may suggest that separate visualisations provide a certain level of information for users, but combining multiple visualisations (in this case, providing graph and tabular data on one screen) would be more powerful and provide more insight to users than using one format alone, in spite of the findings by Prasad and Ohja (2012).

Initially, the researcher believed it would be of use to commuters to provide information on each journey using the tabular visualisation; by displaying on time or early trains in green and late trains in red, the researcher thought that users could identify patterns in arrival times and come to their own conclusions using this data. The majority of testers, however, felt that the data was "*meaningless*" or "*irrelevant*", and that an aggregation of this data would be much more helpful to users. It was revealed that while data relating to individual journeys should remain on the graph visualisation, as it allowed display of groupings of arrival times, as well as identification of outlier journeys that would impact upon average arrival times, the tabular data should only show summaries at the top level. Tabular visualisations, however, should provide a way of extracting daily information on a secondary level; this prototype was modified to allow a linked report from the main tabular visualisation, detailing each journey and its outcome.

5.2.2. One-off journeys

Tourists or local residents who rarely use trains must currently take at face value the published schedule for their selected journeys. Should the train they have caught be delayed by circumstances due to force majeure, there is little that can be done about this; forecasting an event that is out of anyone's control is impossible. However, by providing the probability that a service will run on time by analysing past performance, rail users planning a one-off journey can make a more informed decision about the train journey that they will select.

During task-based testing, it was noted that some testers believed that trains that covered more distance would prove less reliable than trains covering shorter distances. It was suggested that graph visualisations would be slightly less useful for people who are making one-off journeys, as people tended to be "*risk-averse*" and would select the earliest possible service anyway. It was also suggested that there would need to be more explanation given on the graph visualisation as to the information displayed; users who have not used the system before may struggle to comprehend the information that was being displayed.

5.2.3. Station traffic

Operations scheduling is one of the most difficult elements of railway operation. Trains are not like buses, planes, or cars, in that while the aforementioned forms of transport are able to take alternate routes in order to reach their destination, and can be diverted around potential barriers, trains can only run on the rails that have been laid down for their operation.

The researcher's initial rationale for providing a station traffic visualisation was that providing railway operators with a list of all traffic through a station over a given time period could aid in many facets of station operation. Information regarding the volume of trains could give station managers an insight into how many employees they need during a shift to cope with commuter traffic; identifying consistently unreliable trains could lead to operators using different platforms for certain trains; trains that are considered more reliable could be offered opportunities to maintain their performance by being signalling through junctions ahead of trains are already running behind schedule.

During testing, a benefit previously unconsidered was identified. Customer service representatives who are positioned at the station, either at ticket hall level or platform level, may be able to use this visualisation on the platform when dealing with commuters; station display boards may only give very limited information, and by having this visualisation available to hand, staff may be able to better answer commuter enquiries about traffic information.

The researcher believes that one key piece of information has been included that would be of genuine use to railway schedulers, and that is the average arrival and departure time of services. The data being used in this project is not accurate to the second, which does not allow a genuine average length of time spent at a station by a train; using more accurate data, station managers could identify how much time each service spends waiting at a station, either for passengers to embark/disembark, or for a green signal to be shown allowing safe passage through (or both). The ability to identify stations and times when extra time is required by controllers to allow for passenger or train movement, could allow schedulers to allocate more time buffer in the train schedule at those stations, which could result in reduced bottlenecks around stations, and more accurate scheduling of train services.

5.3. Issues uncovered by task-focused evaluation

This section covers issues raised by testers during the task-based evaluation that either were not, or could not, be implemented in the final prototype system, and that could be implemented as part of a more robust solution. Elements that were identified in testing that were not implemented in the

final prototype are listed in bullet points below, and could form the basis for future work (expanded upon in Section 6.2).

- *Display of platform information / direction of train on station traffic report*

During the design phase, it was decided not to capture platform information as the relevance of this to commuters examining historic performance was questioned. While there were no comments regarding the lack of platform information on the visualisations relating to information for commuters, the station traffic visualisation received several comments that platform information would have been useful to determine traffic flow through platforms. Railway staff at stations could use the station traffic visualisation to identify potential bottlenecks with consistently unreliable trains, and identify more suitable platforms for these services to be allocated. That is, of course, only an option at stations with multiple platforms; stations with only two platforms or a single island platform are unlikely to have the capacity to solve these potential issues.

- *Find trains by arrival time at destination*

The prototype was designed such that users could select the time that they wished to depart from their origin station. However, the issue of selecting trains by arrival time at their destination was raised by a couple of testers, who suggested that users may be more interested in deciding what time they want to catch their train by entering their desired arrival time and asking the system to calculate the ideal departure time from their starting point.

Due to time constraints, this feature was not implemented. However, the researcher believes that allowing users would find the option of selecting their arrival time much more useful than choosing their starting time.

- *Providing alternative services or routes for commuters*

One tester suggested that the system should provide alternative trains for commuters to their selected service, should only one or two results be returned from their search. The suggestion was that the train before and the train after (if they were available) should be shown to give commuters extra flexibility when deciding upon their journey.

Alternatively, by using a modified version of the route planner that was investigated in this project, users could be prevent with alternative routes for their journey, if it were to be discovered that users could arrive at their destination quicker due to trains leaving at a more convenient time, or fast trains being available at the same time on a different route.

- *The ability to select only railway stations or all timing points*

The prototype system only shows train stations on any visualisation. A suggestion was made that an option should be made available for railway engineers and scheduling staff to see all timing points along the journeys, not just each calling point, could allow railway staff to identify timing issues at more specific points on each route.

5.4. Recommendations

While this prototype can be used to provide users with information regarding train punctuality, a more robust solution must be constructed in order to satisfy the majority of users, especially if this system were to be deployed online. Additional data capture of historic performance would be essential to provide better results for users; optimised database performance using indexing of tables and possibly the use of database technology more suited to large-scale applications should be investigated.

Should designers of railway information visualisations proceed with graph-based formats, there are several elements to take into account. Firstly, users who are not familiar with Marey-style charts will need some form of explanation or legend as to how they work; this must be provided in a manner that make obvious what the information is attempting to convey, otherwise users will probably decide not to use this information in favour of tabular information that they may decide provides more certainty, despite the fact that the same information is being offered.

With map-based visualisations becoming more prevalent, visualisation designers could look at presenting the information generated in this project using maps, as that may provide a more intuitive result to users. With more accurate journey mapping software, designers may discover that being able to plot train punctuality in a similar format to road traffic data such as (Dijkstra et al., 2001, pg. 31 - 34, see Figures 11, 13 and 14 for examples).

With flexible working hours becoming more accepted throughout business, people may choose to work different hours each week. Using this system would allow users who are regular commuters, but not necessarily at the same time each day, to identify their ideal train for their circumstances, and not rely directly upon the provided schedule.

The researcher is aware that providing information in visual format would not necessarily be accepted by visually impaired railway commuters; while screen readers would have no problem with tabular visualisations, the graph visualisation might prove challenging to any form of technology relating to converting visual to speech. Alternative forms of imparting information should also be investigated.

6. Conclusion / Further Work

6.1. Conclusion

This research project has offered a prototype system that allows users to identify performance reliability concerns with their chosen rail journeys using National Rail trains. This prototype used data modelling techniques to combine schedule information and historic data regarding train activity, and visualisation techniques that display performance information on rail performance using two visualisation formats, tabular and graph formats.

Users are then offered the option to use either or both visualisation formats that they prefer in order to make choices regarding their journeys. Users are also offered the option to view traffic information for a single station over a selected time period, which may prove to be of more use to railway employees than commuters.

Both visualisation formats have their strengths and weaknesses for railway users. Tabular visualisations are more familiar to users and they show aggregated data in a way that users who are only interested in a binary outcome (*"shall I catch this train or not?"*) to come to a faster decision. However, graph visualisations allow users to identify anamolies in their selected services in a way not available to tabular visualisations; a train that has 10 journeys and an average arrival time of two minutes later than scheduled, could be impacted upon by a single journey delayed by 20 minutes. A graph visualisation will inform users far better than a tabular visualisation with solely aggregated data.

Railway commuters and staff members can use this information to make more accurate assessments on the probable reliability of their rail journeys, as well as anticipated traffic at stations and the predicted punctuality of traffic. Using this information could lead to more optimal choices by railway commuters and better scheduling of trains by railway operators, thus allowing a more consistent spread of commuter usage across the network, and reducing the possibility of delays due to passenger embarking and disembarking.

6.2. Future work

This research attempted to illuminate the benefits of providing visualisations regarding historic performance to railway passengers and operators alike. While the researcher feels that this has been achieved in this project, additional work that can be based upon this research has been identified.

The researcher is concerned that users may be presented with information regarding their regular journey and react negatively towards this information; users who discover that their regular

commute is actually more reliable than anticipated may feel resentful towards railway operators, while commuters who are informed that a different journey is more suitable for them may feel that they have an increased freedom of choice due to this new information available to them. It is extremely important that the best way of engaging commuters with the information presented is found, such that commuters are not simply reinforcing their mental models of their selected journeys. To quote Pascal in (2001), *"All men are almost led to believe not of proof, but by attraction"*.

One of the major issues the researcher faced was the quality of data available. As an example, train arrival and departure times were only reported to the nearest minute. For commuters, this may not be a major problem (as people tend to think in minutes and not seconds), but for precision, the ability to access more precise data relating to train arrivals and departures is paramount for this research to be extended. The ability to directly access railway timings would be of greater benefit in situations where aggregation is being used.

An initial aim of this project was to produce visualisations for journeys including at least one change of service. This was not completed due to time constraints, but initial code was built to determine a path between any two stations (if a path exists). This would form additional work towards a much more robust system whereby users could enter their entire journey, regardless of the number of service changes, and see the probability that each connection would arrive in time to meet the next service.

The ability to isolate transit through a particular platform would also be of great benefit to this system. Platform information was considered but ultimately rejected from this project on the grounds that it would not prove significant enough. However, during the task-based testing segment of this project, it was reported by users that capturing this data would have greatly enhanced the usability of the station traffic visualisation.

As mentioned in the Discussion section, this research captured data from trains run over the summer months in the United Kingdom, a time when the weather is at its best for railway operation. It is suggested that collecting this data over a twelve month period would produce much different results, given the seasonality of weather-related delays and cancellations, and provide additional insights into locations where trains struggle to maintain reliability.

Finally, an area that this research did not investigate was the motivations behind why railway users currently choose their journeys. In the researcher's opinion, it is entirely possible that when presented with information regarding their regular journey, users may continue to choose that

service even though it is not optimal for their requirements. For example, people who place a higher value of their family life, or those who are disengaged with their employment, may choose a journey that means they arrive at work slightly late, to gain extra time with their families.

6.3. Personal reflection

The researcher decided upon the project to answer one personal question, common to many commuters; **"why does it feel like my train is never on time?"**

The researcher believes that this prototype system, or at least the ideas surrounding the prototype, could form the basis for a more robust, more user-friendly system that could offer users a much richer option for selecting their rail journey. The researcher believes that this system could be used as the basis for other railway networks, both domestic and international, to offer their commuters a system of selecting their journeys.

The one concern that the researcher does have is that people would use this system to seek validation and possible compensation from railway operators for suboptimal performance, rather than change their behavioural patterns for selecting a train. It is easier to blame others for poor performance rather than change our own patterns; in a world that is not ideal, the research does not believe that we need another stick with which to beat organisations dealing with multiple failures in past planning; rather, by collecting and analysing data, and providing it to commuters and operators alike in an easy to comprehend format, small changes can be made to provide large-scale benefit to all.

References

Allport, D.A., 1971. Parallel encoding within and between elementary stimulus dimensions. Percept. Psychophys. 10, 104–108. doi:10.3758/BF03214327

Ayhan, S., Pesce, J., Comitz, P., Sweet, D., Bliesner, S., Gerberick, G., 2013. Predictive analytics with aviation big data. IEEE, pp. 1–13. doi:10.1109/ICNSurv.2013.6548556

Chymera, M., Goodman, C.J., 2012. Overview of electric railway systems and the calculation of train performance. Institution of Engineering and Technology, pp. 1–18. doi:10.1049/ic.2012.0070

Dijkstra, J., Timmermans, H.J., Jessurun, A., 2001. A multi-agent cellular automata system for visualising simulated pedestrian activity, in: Theory and Practical Issues on Cellular Automata. Springer, pp. 29–36.

Friel, S.N., Curcio, F.R., Bright, G.W., 2001. Making Sense of Graphs: Critical Factors Influencing Comprehension and Instructional Implications. J. Res. Math. Educ. 32, 124. doi:10.2307/749671

Goverde, R.M., 2005. Punctuality of railway operations and timetable stability analysis. Netherlands TRAIL Research School.

Goverde, R.M., Odijk, M.A., 2002. Performance evaluation of network timetables using PETER. Comput. Railw. VIII 731–740.

Healey, C.G., 1996. Choosing effective colours for data visualization. ACM, pp. 263–270,. doi:10.1109/VISUAL.1996.568118

Kanai, S., Shiina, K., Harada, S., Tomii, N., 2011. An optimal delay management algorithm from passengers' viewpoints considering the whole railway network. J. Rail Transp. Plan. Manag. 1, 25–37. doi:10.1016/j.jrtpm.2011.09.003

Kazman, R., Carriere, J., 1996. Rapid prototyping of information visualizations using VANISH. IEEE Comput. Soc. Press, pp. 21–28,. doi:10.1109/INFVIS.1996.559212

Larman, C., Basili, V.R., 2003. Iterative and incremental development: A brief history. Computer 36, 47–56.

Lujin Wang, Giesen, J., McDonnell, K.T., Zolliker, P., Mueller, K., 2008. Color Design for Illustrative Visualization. IEEE Trans. Vis. Comput. Graph. 14, 1739–1754. doi:10.1109/TVCG.2008.118

Nielsen, J., 1994. Usability inspection methods. ACM Press, pp. 413–414. doi:10.1145/259963.260531

Nielsen, J., 2000. Why You Only Need to Test with 5 Users.

Oates, B.J., 2006. Researching information systems and computing. Sage.

Office of National Statistics, 2013. Census reveals details of how we travel to work in England and Wales. Office of National Statistics.

Office of Rail Regulation, 2014. Public performance measure (PPM) moving annual average (MAA) by sector.

Pascal, B., 2001. De l'art de persuader. Éd. Mille et une nuits, Paris.

Prasad, G.V.R.J.S., Ojha, A., 2012. Text, Table and Graph -- Which is Faster and More Accurate to Understand? IEEE, pp. 126–131. doi:10.1109/T4E.2012.18

Raper, J., 2011. Why train departure information is not currently open data. TransportAPI.

Rehm, F., Klawonn, F., Russ, G., Kruse, R., 2007. Modern Data Visualization for Air Traffic Management. IEEE, pp. 19–24. doi:10.1109/NAFIPS.2007.383804

Rougeux, N., 2011. Visualizing Metra.

Rubin, J., Chisnell, D., 2008. Handbook of usability testing how to plan, design, and conduct effective tests. Wiley Pub., Indianapolis, IN.

Santiago, C., Rusu, A., Falconi, L., Petzinger, B., Crowell, A., 2009. Overview and flight-by-flight analysis of trajectory prediction systems using a 2D Galaxy Visualization. IEEE, pp. 3.B.3–1–3.B.3–8. doi:10.1109/DASC.2009.5347524

Simunovic, M.P., 2010. Colour vision deficiency. Eye 24, 747–755. doi:10.1038/eye.2009.251

Stanford Visualisation Group, 2010. Marey's Trains.

Tufte, E.R., 2001. The visual display of quantitative information, 2nd ed. ed. Graphics Press, Cheshire, Conn.

Yao, R., Jiandong, W., Jianli, D., 2009. RIA-based visualization platform of flight delay intelligent prediction. IEEE, pp. 94–97. doi:10.1109/CCCM.2009.5267976

Zhong, Y., 2008. Study on Cognitive Decision Support Based on Learning and Improvement of Mental Models. IEEE, pp. 490–494. doi:10.1109/CCCM.2008.148

APPENDICES

Section 1. Research Proposal

Section 2. System Code

Section 3. Data Examples (Schedule and Performance)

Section 4. Database Design (Entity-Relationship Diagram and Relational Model)

Section 5. Evolution of Visualisations

Section 6. Task-Based Testing Feedback

APPENDIX 1: RESEARCH PROPOSAL

1. Introduction

Worldwide, people travel to and from their homes to work or other destinations via public transportation. As an example, in the United Kingdom in 2011, five percent of work commuters travel via the National Rail-owned railway system (Office of National Statistics, 2013). Many of these commuters will make multi-leg journeys, which is those that consist of at least one change of train.

The existing journey planners that exist for commuters, however, consist primarily of numeric-based information presented in tabular format, with connection information restricted to options based on fixed schedules provided by train company operators. In a world where trains were completely reliable, this would be sufficient for any commuter to make the best possible choice. It is not true, however, that we live in a perfect world; according to performance figures issued up to the end of the 3rd quarter for 2013/14 for the United Kingdom, one in ten trains were to arrive at their destination more than five minutes late (Office of Rail Regulation, 2014).

For commuters who must arrive at their destination at a given time, or must catch a connecting train, a 10% probability that a train does not arrive on time would be an unacceptable outcome to their journey. Representing information in new ways can give commuters the opportunity to make better decisions regarding their travel patterns.

The research in this proposal will attempt to provide an answer to the following question:

How can data modelling and visualisation techniques be used to provide rail commuters with more informed choices?

The purpose of this research is to determine whether visualisation techniques can be used effectively in conjunction with data modelling, to create representations of information that provide rail commuters with more intuitive information to plan their journey.

At the end of this research project, the aim is to produce two definitive outputs;

4. A pair of data sets; one that contains a full published list of National Rail scheduling, and one that contains the actual performance of each train, including system cancellations, alterations, and planned changes, and
5. A set of visualisations that provides analysis of actual railway performance compared to the published schedule of train operations, with the aim of providing additional information to commuters about their travel needs.

The ideal outcome for this research is to provide rail commuters with a more informed choice as to their travelling habits, and allow rail operators to more actively engage with commuters about their travelling choices.

This research will focus solely on trains that are run by National Rail; those trains run under the auspices of metropolitan transit providers such as Transport for London (for London Underground and Overground services), as well as public transport services such as buses, trams, or private transportation options, are not included in this research, although it is foreseeable that further research could include these transportation services. The area of schedule operation and forecasting is outside the scope of this research, as is any potential commercial usage of this research.

2. Critical Context

Existing information that is provided for rail commuters, at least in the United Kingdom, is largely in the form of point-to-point timetables (fixed by the railway operators), allowing little flexibility in

decision making processes; commuters can only rely on the published schedule and their own empirical experience to determine their travel patterns. This can lead to faulty mental models being constructed by commuters, which can lead to a "*powerful impact on … framing behaviour*" of the decisions of commuters (Zhong, 2008).

In the field of evaluating the reliability of trains, an important piece of research comes from Goverde and Odijk (2002), and their work on evaluating the robustness of Dutch railway network timetables using an analytical tool that measures the published timetables against performance, with a case study that uses a stand-alone analysis tool to depict the Dutch Intercity Network with any potential delay propagation throughout the network (2002, figures 1 and 2). Goverde extended his work in 2005 as part of his PhD thesis on railway punctuality and timetable stability (2005), incorporating his comparison of published schedule information to the actual performance of trains. Goverde goes into significant detail when describing the Dutch railway system, making clear the importance of correct timetable scheduling for the network. The case study regarding punctuality of trains using Eindhoven railway station (2005, chapter 5, pg. 97-122) goes into great detail with his analysis into the reasons behind punctuality, a section that will inform this proposed research. Finally, figure 8.7 (2005, pg. 255) continues his work from (2002) by showing the impact of delays upon the network, and how much recovery time is available to the train should it be subject to delay.

Goverde's research follows a highly quantitative path (his data analysis, combined with application of algorithms) and appears to have more concern with the optimization of timetables; this proposed research aims to combine a quantitative approach with a qualitative outcome (data visualisation). Goverde admits that "*future effort must be directed towards usage of the empirical data in daily practice*" (2005), something that this research aims to provide, although in this case it will be for commuters, rather than the Dutch railway network that Goverde goes on to discuss.

While the visualisation of real-time information appears uncommon in rail transportation literature, with the focus largely remaining on the scheduling of trains, the aviation industry provides a rich seam of literature that can be used as a starting block for information on the visual representation of data. Examples of this include Yao and Jiandong's research into a platform that can be used to predict delays for airports and airlines (Yao et al., 2009), using multi-dimensional scaling plots to flight durations and the effect of wind speed and visibility on traffic demand (Rehm et al., 2007, pp. 22–23), and determining the trajectory of aircraft using a bespoke visualisation tool (Santiago et al., 2009). While the nature of data logging in aviation cannot be completely replicated in the area of railways, the basic structure of rail data can be considered to be similar to those in aviation; primarily, a unique identifier for the route, its origin, its destination, expected and actual departure and arrival times, its current location, and any reasons for delays or cancellations.

An additional idea brought from research into aviation data comes from (Ayhan et al., 2013) and their work into data warehousing and tracking of real-time data. They use ASDI[7] information sent by the Federal Aviation Administration in the United States to analyse make "*predictions based upon descriptive patterns of massive aviation data*". Their work with *"operational real-time or near real-time surveillance data"* describes their methodology in using the ASDI information in conjunction with real-time tracking of planes. The architecture structure for converting raw real-time data into a usable format for analysis in (Ayhan et al., 2013, pp. I8–6) provides a basis for this research; namely, the capture of real-time data, correlated with existing scheduling information, to provide one combined working set of information, that can be used for the visualisation section of this research.

Inspiration for this research is also drawn from the work of E. J. Marey and his graphical train schedule (first published in 1878 and reproduced on the cover of (Tufte, 2001)). Considered ground-breaking at the time, the visualisation technique has been implemented using contemporary source

[7] Aircraft Situation Display to Industry

data (examples by (Rougeux, 2011) and (Stanford Visualisation Group, 2010)). While the visualisation technique used is easy to follow, especially with modern techniques that use different colours to distinguish between routes, a limitation of this visualisation is that it shows an "ideal world" in which trains run on time 100% of the time. According to National Rail data, trains in the United Kingdom meet their performance targets only 90% of the time (Office of Rail Regulation, 2014). With the advent of real-time data being made available, this research will look to extend Marey's visualisation concept by combining existing scheduling data for rail in the United Kingdom and historical railway performance.

Marey's graphical representation appears to influence a small part of the work of Kanai et al. (2011). In their research, Kanai et al. propose an algorithm for delay management of trains in the Japanese railway system, using a traffic simulator to forecast traffic flows using their algorithm. Figure 7 shows how the combination of timetable data and real-time data is used to influence their simulator design, while Figure 10 shows the behaviour model of a passenger's decision making process. Where this research aims to provide its purpose is before the "Appear at Station" action in Figure 10; by making information available before a user arrives at the station as to the likelihood of their train arriving on time, users can make more informed choices for their journey (though, as noted in Kanai et al., the possibility exists that not every passenger makes their choice based solely on quantitative information).

3. Methods & Tools for Analysis & Evaluation

This research will use two distinct research methods to answer the posed question. Firstly, as this research will use quantitative data analysis (Oates, 2006, p. 38), a new system will be built in order to take the existing data set and transform it into a data set that will allow more investigation to take place. Secondly, because this research will attempt to create visualisations that require users to interpret information in a way that is both easily understood and aesthetically pleasing, qualitative data analysis (Oates, 2006, p. 38) will be deployed in the second phase of the project. The combination of the two research methods is essential as while the existing data needs to be transformed into meaningful information, the layout of that information will convey information to the end user in a format that provides the most critical information possible as quickly as possible.

Several sets of transport information are made available to developers that are relevant to this research. National Rail provide real-time information regarding train movement, signal information, delay and cancellation advice, and performance measurement. The Association of Train Operating Companies (ATOC) make available scheduling information for every major train (including passenger and freight) that operates in the mainland United Kingdom, as well as station locations and additional transit information between stations via other methods (walk, bus, Underground, etc.). A capture of live data is already being undertaken to ensure the most amount of available data is available for use.

The most appropriate method for the manipulation and output of the raw data stream to convert this data into a database format. The existing data stream for the railway schedule is produce in a Common Interface File (CIF) format, which has a header, data, and terminating field for each train, station, and link that is in use, but this format is very difficult to produce meaningful information from. As the two streams of data are released separately (scheduling information is static with occasional updates, while actual information is sent close to real-time), this research plans to use a combination of an internet scripting language and a database server, similar to the concepts raised by Yao and Jiandong (Yao et al., 2009, p. 95), to capture the real-time data from National Rail, and convert the scheduling data from CIF format into a format that is better suited to databases.

The database will be constructed first by analysing the input provided by the raw data streams. As mentioned in the previous paragraph, schedule data is provided in CIF format; real-time data is

provided via the JSON messaging system. This phase will require attempting to identify all common data fields between the two sources, such as train identifier or schedule code. After that is completed, an entity relationship diagram (ERD) is to be design, in order to determine the overall physical structure of the database, and to determine the primary and foreign keys for the tables to be created. A relational model will be generated from this ERD, to provide full documentation of the design specification for the purposes of traceability (Oates, 2006, p. 113) and replication.

The next step is to design the data parser that will be used to convert the raw data into a usable data format. Using a series of UML diagrams to show the programmatic flow will provide traceability to the final system, as well as provide the blueprint for the implementation. The parser will then be implemented using an internet scripting language that will then feed directly into the previously created database. The most likely choice of language at this stage is PHP, as this is the language that the researcher is most familiar with. The choice of language also influences the choice of database; the most likely database programme to be chosen is MySQL, for the same reason as the choice of PHP. Unit testing of procedures will take place to ensure that the expected programme flow from the design phase is being followed in the implantation. This will entail the usage of a framework such as PHPUnit[8], and will be carried out for all code written for the parser.

An integration tool must be designed and implemented in order to combine the results from the parser. Firstly, the results from the parser output will be analysed to determine the structure for the data; UML diagrams will then be used to design the programme flow, and the tool will be implemented using the same internet scripting language as the parser. Thus, the final system will therefore include the following components: a parser, an integration tool, and two databases.

The visualisation output will most likely be produced using the Processing programming language[9]. This language is made available through Open Source, reducing concerns about the usage of proprietary software. The Processing language provides additional flexibility in controlling the desired output for users, can be used by non-visual programmers (such as this researcher), and offers additional tools that can be used to produce complex visualisations. Alternative software solutions will be investigated should this language prove to be unsuitable during the course of the qualitative research phase; candidates for this include HighCharts[10], GoogleCharts[11] for internet-based visuals, and JavaFX[12], a solution capable of desktop and internet-based outputs. A prototype visualisation will be built initially, as this will "*encourage interactive exploration of different implementation possibilities*" (Kazman and Carriere, 1996, p. 1). This prototype will then refined as the development cycle continues its path, until a stable solution is decided upon.

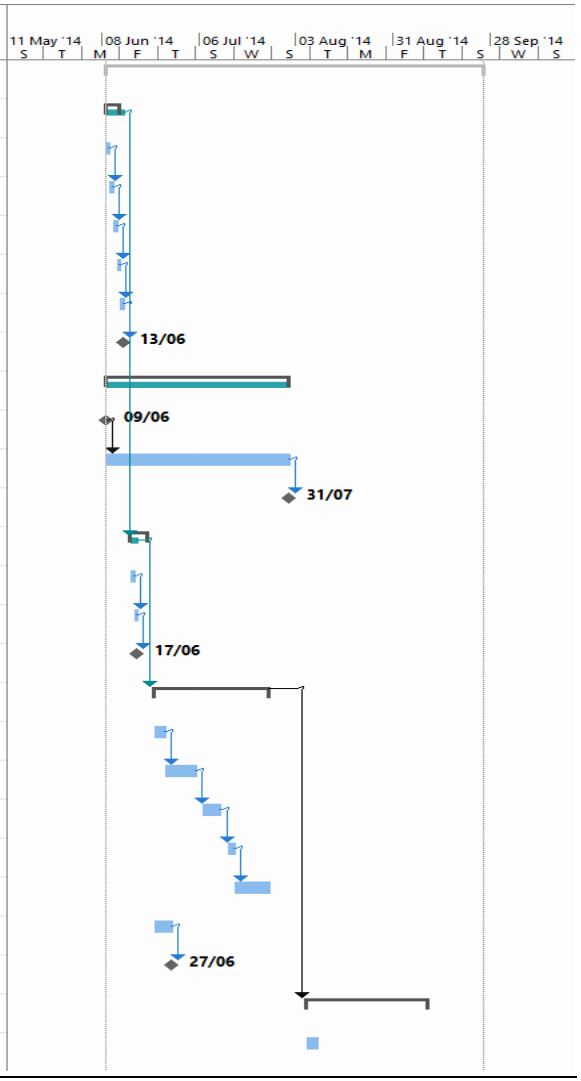
To conduct the most appropriate evaluation of the visualisation output, heuristic evaluation (Nielsen, 1994) will be conducted in order to find shortcomings in the output. At least one expert will be engaged to follow a strict set of guidelines to determine whether there are any glaring omissions or inclusions in the visualisations produced. Once their testing has been completed, usability testing will be performed (Rubin and Chisnell, 2008) with a select group of commuters whose opinions would be sought on the visualisations that have been generated.

The final evaluation will then determine whether the aims, objectives, and deliverables of the research were achieved, and whether an answer to the posed research question was achieved.

[8] http://phpunit.de/
[9] http://processing.org/
[10] http://www.highcharts.com/
[11] https://developers.google.com/chart/
[12] http://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html

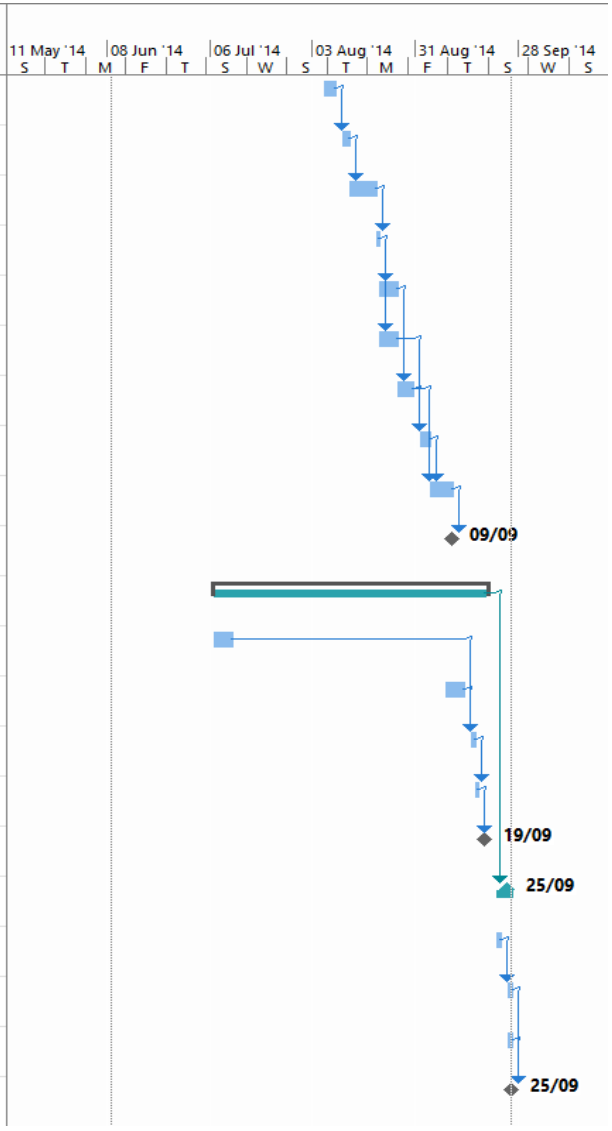
4. Work Plan

ID	Task Name	Duration	Start	Finish	Predecessors
0	**Model and Visualisation Development**	**79 days**	**Mon 09/06/**	**Thu 25/09/1**	
1	**Write parser to interpret schedule data**	**4 days**	**Mon 09/06/14**	**Thu 12/06/14**	
2	Analyse schedule data from CIF file	1 day	Mon 09/06/14	Mon 09/06/14	
3	Design of parser	1 day	Tue 10/06/14	Tue 10/06/14	2
4	Implementation of parser using PHP to organ	1 day	Wed 11/06/14	Wed 11/06/14	3
5	Conduct unit testing	1 day	Thu 12/06/14	Thu 12/06/14	4
6	Run data through parser	1 day	Fri 13/06/14	Fri 13/06/14	5
7	Parser complete	0 days	Fri 13/06/14	Fri 13/06/14	6
8	**Capture of live data feed (already in progress)**	**39 days**	**Mon 09/06/14**	**Thu 31/07/14**	
9	Set up capture of data feed	0 days	Mon 09/06/14	Mon 09/06/14	
10	Automatic data capture	39 days	Mon 09/06/14	Thu 31/07/14	9
11	Capture of live data complete	0 days	Thu 31/07/14	Thu 31/07/14	10
12	**Analysis of scheduling data**	**5 days**	**Mon 16/06/14**	**Fri 20/06/14**	**1**
13	Ensure that data structure is appropriate	1 day	Mon 16/06/14	Mon 16/06/14	
14	Ensure that no data has been lost or badly co	1 day	Tue 17/06/14	Tue 17/06/14	13
15	Scheduling data analysis complete	0 days	Tue 17/06/14	Tue 17/06/14	14
16	**Design and testing of integration tool**	**25 days**	**Mon 23/06/14**	**Fri 25/07/14**	**12**
17	Design of integration tool	3 days	Mon 23/06/14	Wed 25/06/14	
18	Build schedule data / performance data integration tool	7 days	Thu 26/06/14	Fri 04/07/14	17
19	Conduct unit testing	5 days	Mon 07/07/14	Fri 11/07/14	18
20	Run initial outputs	2 days	Mon 14/07/14	Tue 15/07/14	19
21	Refine integration tool based on initial outpu	8 days	Wed 16/07/14	Fri 25/07/14	20
22	Complete black box testing	5 days	Mon 23/06/14	Fri 27/06/14	
23	Testing complete	0 days	Fri 27/06/14	Fri 27/06/14	22
24	**Visualisation Development**	**25 days**	**Wed 06/08/14**	**Tue 09/09/14**	**16**
25	Develop initial visualisation prototype using data from model	3 days	Wed 06/08/14	Fri 08/08/14	



(cont. on next page)

ID	Ta M	Task Name	Duration	Start	Finish	Predecessors
25		Develop initial visualisation prototype using data from model	3 days	Wed 06/08/14	Fri 08/08/14	
26		Conduct initial evaluation	2 days	Mon 11/08/14	Tue 12/08/14	25
27		Refine visualisation template	5 days	Wed 13/08/14	Tue 19/08/14	26
28		Approve template	1 day	Wed 20/08/14	Wed 20/08/14	27
29		Develop materials for heuristic evaluation	3 days	Thu 21/08/14	Mon 25/08/14	28
30		Develop materials for usability testing	3 days	Thu 21/08/14	Mon 25/08/14	28
31		Conduct heuristic evaluation	4 days	Tue 26/08/14	Fri 29/08/14	29
32		Conduct usability testing	3 days	Mon 01/09/14	Wed 03/09/14	30,31
33		Interpret data from evaluation and testing	4 days	Thu 04/09/14	Tue 09/09/14	32,31
34		Visualisation development complete	0 days	Tue 09/09/14	Tue 09/09/14	33
35		**Document outcomes**	**55 days**	**Mon 07/07/14**	**Fri 19/09/14**	
36		Document findings from model developmen	5 days	Mon 07/07/14	Fri 11/07/14	
37		Document findings from visualisation evalua	5 days	Mon 08/09/14	Fri 12/09/14	
38		Document conclusions and further research	1 day	Mon 15/09/14	Mon 15/09/14	36,37
39		Complete abstract and introduction	1 day	Tue 16/09/14	Tue 16/09/14	38
40		Documentation complete	0 days	Fri 19/09/14	Fri 19/09/14	39
41		**Make Final Submission**	**0 days**	**Thu 25/09/14**	**Thu 25/09/14**	**1,8,12,16,24,35**
42		Print and bind final submission	1 day	Mon 22/09/14	Mon 22/09/14	
43		Submit hardcopy at university	1 day?	Thu 25/09/14	Thu 25/09/14	42
44		Submit softcopy through Moodle	1 day?	Thu 25/09/14	Thu 25/09/14	
45		Submission complete	0 days	Thu 25/09/14	Thu 25/09/14	44,43



5. Risk Register

During any attempt to conduct research, the possibility exists that issues beyond the control of the researcher may occur. The register below is an attempt to identify all risks that can be understood at this stage of research, their likelihood and impact upon the outcomes of the research, and a proposed mitigation strategy to prevent these risks either happening, or having a material impact upon the successful completion of this research.

Identified Risk	Likelihood	Impact	Mitigation Strategy
Withdrawal of data feed by National Rail	Low - Medium	Low to severe, depending on the stage of the project	Capture as much data as possible before project commencement; identify alternate sources in the case of feed withdrawal.
Terms of usage are changed by data provider	Low	Low to moderate depending on the level of change in the TOC	Ensure that all research is conducted to comply with the TOC; evaluate outcomes if the TOC changes to prevent or curtail level of data provided
Server or system failure resulting in loss of research information	Low	Low to catastrophic, depending on the stage of the project	All information, notes, and data to be backed up daily to off-site backup facility to reduce potential impact
Inability to recruit experts to complete heuristic testing	Low – Medium	Moderate	Identify and make contact with experts as quickly as possible to ensure availability
The graphical language chosen (Processing) proves unsuitable	Medium	Very low as alternatives can be sought	Ensure that a base level of knowledge is established before project commencement
Tasks taking longer to complete than estimated on work plan	Low	Moderate to severe, depending on stage of project and length of overrun	Work plan revisions will be made as tasks are completed. Slight changes to original plan are anticipated due to the nature of project work; major changes will require re-drafting of work plan and possible discussion with supervisor
Participants inadvertently identified during research	Low	Complete breach of participant confidentiality	Any identifying work to be removed immediately from research. Breach to be reported promptly to supervisor.

6. Ethical, Professional & Legal Issues

It is desirable to conduct research in such a manner that there is no possibility of ethical, professional, or legal controversy occurring. When consulting the Research Ethics Checklist as provided by City University, there are questions that require additional exploration to ensure that no controversy arises from this research.

A question relates to the consent of any participants being obtained before their engagement. A consent form has been drawn up that all participants will be required to sign before their assistance with this research can be evaluated or incorporated. It is expected that this will take place during the qualitative analysis phase of this research, as it is desirable that experts in visualisation techniques, user interface layout, and railway focus groups will be brought in to undertake a heuristic evaluation of the output produced by this research (Nielsen, 1994). Their identities will remain confidential unless their express approval is given to identify them if their findings markedly influence the outcome of this research.

In this case of usability testing that will be conducted with a sample set of users who it is believed would benefit from this research, their identities will remain confidential, with their consent obtained before any testing is conducted, and their answers will only be distinguished using a generic letter/number code. A separate form has been drawn up for them to sign, as their involvement in the research will be different from those participants in the heuristic testing.

The researcher has noted that research may be conducted with participants when they are not inside the boundaries of City University. It must be noted that every attempt will be made to engage with participants at City University; there may be circumstances that participants cannot attend the university to offer their assistance, and in those cases a neutral venue to both researcher and participant will be decided upon.

The data providers listed above require adherence to their Terms and Conditions that state (amongst other conditions) that data can be manipulated for display purposes, but not amended. This research will make every attempt to remain within the boundaries of all Terms and Conditions at all times, as provided fraudulent data would provide commuters and railway operators with inaccurate information. A set of invented data will be generated for the purposes of the testing phase of any system builds, and will be deleted for any reports generated by the final system.

References

Allport, D.A., 1971. Parallel encoding within and between elementary stimulus dimensions. Percept. Psychophys. 10, 104–108. doi:10.3758/BF03214327

Ayhan, S., Pesce, J., Comitz, P., Sweet, D., Bliesner, S., Gerberick, G., 2013. Predictive analytics with aviation big data. IEEE, pp. 1–13. doi:10.1109/ICNSurv.2013.6548556

Chymera, M., Goodman, C.J., 2012. Overview of electric railway systems and the calculation of train performance. Institution of Engineering and Technology, pp. 1–18. doi:10.1049/ic.2012.0070

Dijkstra, J., Timmermans, H.J., Jessurun, A., 2001. A multi-agent cellular automata system for visualising simulated pedestrian activity, in: Theory and Practical Issues on Cellular Automata. Springer, pp. 29–36.

Friel, S.N., Curcio, F.R., Bright, G.W., 2001. Making Sense of Graphs: Critical Factors Influencing Comprehension and Instructional Implications. J. Res. Math. Educ. 32, 124. doi:10.2307/749671

Goverde, R.M., 2005. Punctuality of railway operations and timetable stability analysis. Netherlands TRAIL Research School.

Goverde, R.M., Odijk, M.A., 2002. Performance evaluation of network timetables using PETER. Comput. Railw. VIII 731–740.

Healey, C.G., 1996. Choosing effective colours for data visualization. ACM, pp. 263–270,. doi:10.1109/VISUAL.1996.568118

Kanai, S., Shiina, K., Harada, S., Tomii, N., 2011. An optimal delay management algorithm from passengers' viewpoints considering the whole railway network. J. Rail Transp. Plan. Manag. 1, 25–37. doi:10.1016/j.jrtpm.2011.09.003

Kazman, R., Carriere, J., 1996. Rapid prototyping of information visualizations using VANISH. IEEE Comput. Soc. Press, pp. 21–28,. doi:10.1109/INFVIS.1996.559212

Larman, C., Basili, V.R., 2003. Iterative and incremental development: A brief history. Computer 36, 47–56.

Lujin Wang, Giesen, J., McDonnell, K.T., Zolliker, P., Mueller, K., 2008. Color Design for Illustrative Visualization. IEEE Trans. Vis. Comput. Graph. 14, 1739–1754. doi:10.1109/TVCG.2008.118

Nielsen, J., 1994. Usability inspection methods. ACM Press, pp. 413–414. doi:10.1145/259963.260531

Nielsen, J., 2000. Why You Only Need to Test with 5 Users.

Oates, B.J., 2006. Researching information systems and computing. Sage.

Office of National Statistics, 2013. Census reveals details of how we travel to work in England and Wales. Office of National Statistics.

Office of Rail Regulation, 2014. Public performance measure (PPM) moving annual average (MAA) by sector.

Pascal, B., 2001. De l'art de persuader. Éd. Mille et une nuits, Paris.

Prasad, G.V.R.J.S., Ojha, A., 2012. Text, Table and Graph -- Which is Faster and More Accurate to Understand? IEEE, pp. 126–131. doi:10.1109/T4E.2012.18

Raper, J., 2011. Why train departure information is not currently open data. TransportAPI.

Rehm, F., Klawonn, F., Russ, G., Kruse, R., 2007. Modern Data Visualization for Air Traffic Management. IEEE, pp. 19–24. doi:10.1109/NAFIPS.2007.383804

Rougeux, N., 2011. Visualizing Metra.

Rubin, J., Chisnell, D., 2008. Handbook of usability testing how to plan, design, and conduct effective tests. Wiley Pub., Indianapolis, IN.

Santiago, C., Rusu, A., Falconi, L., Petzinger, B., Crowell, A., 2009. Overview and flight-by-flight analysis of trajectory prediction systems using a 2D Galaxy Visualization. IEEE, pp. 3.B.3–1–3.B.3–8. doi:10.1109/DASC.2009.5347524

Simunovic, M.P., 2010. Colour vision deficiency. Eye 24, 747–755. doi:10.1038/eye.2009.251

Stanford Visualisation Group, 2010. Marey's Trains.

Tufte, E.R., 2001. The visual display of quantitative information, 2nd ed. ed. Graphics Press, Cheshire, Conn.

Yao, R., Jiandong, W., Jianli, D., 2009. RIA-based visualization platform of flight delay intelligent prediction. IEEE, pp. 94–97. doi:10.1109/CCCM.2009.5267976

Zhong, Y., 2008. Study on Cognitive Decision Support Based on Learning and Improvement of Mental Models. IEEE, pp. 490–494. doi:10.1109/CCCM.2008.148

Research Ethics Checklist

School of Informatics BSc, MSc, MA Projects

If the answer to any of the following questions (1 – 3) is NO, your project needs to be modified.	*Delete as appropriate*
1. Does your project pose only minimal and predictable risk to you (the student)?	**YES**
2. Does your project pose only minimal and predictable risk to other people affected by or participating in the project?	**YES**
3. Is your project supervised by a member of academic staff of the School of Informatics or another individual approved by the module leaders?	**YES**

If the answer to either of the following questions (4 – 5) is YES, you MUST apply to the University Research Ethics Committee for approval. (You should seek advice about this from your project supervisor at an early stage.)	*Delete as appropriate*
4. Does your project involve animals?	**NO**
5. Does your project involve pregnant women or women in labour?	**NO**

If the answer to the following question (6) is YES, you MUST complete the remainder of this form (7 – 19). If the answer is NO, you are finished.	*Delete as appropriate*
6. Does your project involve human participants? For example, as interviewees, respondents to a questionnaire or participants in evaluation or testing?	**YES**

If the answer to any of the following questions (7 – 13) is YES, you MUST apply to the Informatics Research Ethics Panel for approval and your application may be referred to the University Research Ethics Committee. (You should seek advice about this from your project supervisor at an early stage.)	*Delete as appropriate*
7. Could your project uncover illegal activities?	**NO**
8. Could your project cause stress or anxiety in the participants?	**NO**
9. Will you be asking questions of a sensitive nature?	**NO**
10. Does your project rely on covert observation of the participants?	**NO**
11. Does your project involve participants who are under the age of 18?	**NO**

12.	Does your project involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)?	**NO**
13.	Does your project involve participants who have learning difficulties?	**NO**
The following questions (14 – 16) must be answered YES, i.e. you MUST COMMIT to satisfy these conditions and have an appropriate plan to ensure they are satisfied.		*Delete as appropriate*
14.	Will you ensure that participants taking part in your project are fully informed about the purpose of the research?	**YES**
15.	Will you ensure that participants taking part in your project are fully informed about the procedures affecting them or affecting any information collected about them, including information about how the data will be used, to whom it will be disclosed, and how long it will be kept?	**YES**
16.	When people agree to participate in your project, will it be made clear to them that they may withdraw (i.e. not participate) at any time without any penalty?	**YES**
The following questions (17 – 19) must be answered and the requested information provided.		*Delete as appropriate*
17.	Will consent be obtained from the participants in your project? *Consent forms made available on the next page.*	**YES**
18.	Have you made arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential? *No information from any participant will be recorded unless consent is provided. All consent forms and records of testing will be sealed and provided to the university as part of the thesis submission. Any information found to have inadvertently contained identifying information will be withdrawn from the research and disposed of using secure shredding.*	**YES**
19.	Will the research be conducted in the participant's home or other non-University location? *This answer is given as YES, even though it is expected that all testing with participants will be conducted at City University, using the equipment of City University. This answer covers the possibility that participants may not be able to undertake activities at City University; a neutral venue will be arranged where testing can take place in this instance. Research will not be conducted at the residence or workplace of participants, except in the case where a City University employee is involved.*	**YES**

RESEARCH CONSENT FORM – HEURISTIC EVALUATION

Name of Researcher
Matthew Griffin
Title of study
Data visualization with rail commuters and railway operators

Please read this form fully and carefully, and complete the form. If you are willing to participate in this study, please note the appropriate responses to all questions, then sign and date the declaration at the end. If you do not understand anything and would like more information, do not hesitate to ask for clarification.

- I have had the research satisfactorily explained to me in verbal and / or written form by the researcher. **YES / NO**
- I understand that the research will involve an interview, the duration of which will take no longer than 1 hour, to be held at City University or at a neutral venue. **YES / NO**
- I understand that I may withdraw my assistance to this study at any time without having to give an explanation. **YES / NO**
- I understand that all information about me will be treated in the strictest confidence, and that I will not be named in any written work arising from this study, unless additional permission is sought to do so. **YES / NO**
- I understand that any material I complete during this study will be destroyed upon the conclusion of this research. **YES / NO**
- I understand that you will be discussing the progress of your research with your supervisor and other staff at City University London. **YES / NO**

I freely give my consent to participate in this research study and have been provided with a copy of this form for my record keeping.

Signature: ……

Date: ……

RESEARCH CONSENT FORM – USABILITY TESTING

Name of Researcher
Matthew Griffin
Title of study
Data visualization with rail commuters and railway operators

Please read this form fully and carefully, and complete the form. If you are willing to participate in this study, please note the appropriate responses to all questions, then sign and date the declaration at the end. If you do not understand anything and would like more information, do not hesitate to ask for clarification.

- I have had the research satisfactorily explained to me in verbal and / or written form by the researcher. **YES / NO**
- I understand that the research will involve a brief interview of no longer than 15 minutes, to be held at City University or a neutral venue should City University not be a suitable location. **YES / NO**
- I understand that I may withdraw my assistance to this study at any time without having to give an explanation. **YES / NO**
- I understand that all information about me will be treated in the strictest confidence, and that I will not be named in any written work arising from this study. **YES / NO**
- I understand that any material I complete during this study will be destroyed upon the conclusion of this research. **YES / NO**
- I understand that you will be discussing the progress of your research with your supervisor and other staff at City University London. **YES / NO**

I freely give my consent to participate in this research study and have been provided with a copy of this form for my record keeping.

Signature: ……

Date: ……

APPENDIX 2: SYSTEM CODE

Page: index.php

```
<?php

/*==================================
FILE: index.php

PURPOSE: To allow users to select options for train journeys or traffic through one
station.

CREDITS: All work by Matt Griffin

==================================*/

//Extend PHP's maximum script execution time in order to process the required data
ini_set('max_execution_time', 6000);

$con = mysqli_connect("localhost","root","","nrdata");

// Check connection
if (mysqli_connect_errno())
{
	echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

//this is a fairly simple index page. there are two links on this page, to allow
users to make their selections
//a more robust system would make this look much prettier but for the purposes of a
prototype this will suffice
echo "<HTML><HEAD>";
echo "<link rel='stylesheet' type='text/css' href='stylesheet.css'>";
echo "<TITLE>Welcome to the Selection Screen</TITLE></HEAD><BODY>";
echo "<CENTER><b>RAILWAY PERFORMANCE - SOUTHERN UK TRAINS</b></center><p>";
echo "<div class='CSSTableGenerator'>";
echo "<table border='3'>";
echo "<tr>";
echo "<td><a href='NRMultiTrainInfo.php'>Select To/From/Via Stations</a></td>";
echo "<td><a href='NRMultiServiceInfoAtStation.php'>Traffic for One
Station</a></td>";
echo "</tr>";
echo "</body></html>";
?>
```

Page: NRFindRoute.php

```
<?php

/*==================================
FILE: NRFindRoute.php

PURPOSE: To determine the shortest path between two stations, whether or not they
are connected
         physically via rail tracks. The code returns the shortest path based upon
the NRFixedLinks
         table; if a path cannot be found, no path will be returned.
         NB: this code is only provided as an example of what has been achieved to
date. This is left
         here to demonstrate what could be achieved with further work.

CREDITS: Ignatius Teo (http://www.sitepoint.com/data-structures-4/) whose code on
shortest-path
         algorithms (the Dijkstra and Graph classes below) has been adapted for use
in this project

==================================*/

ini_set('max_execution_time', 6000);

$con = mysqli_connect("localhost","root","","nrdata");

// Check connection
if (mysqli_connect_errno())
{
       echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

$today = date("Y-m-d");

error_reporting(E_ALL & ~E_NOTICE);

require("vendor/autoload.php");

class Graph
{
       protected $graph;
       protected $visited = array();

       public function __construct($graph) {
              $this->graph = $graph;
       }

       // find least number of hops (edges) between 2 nodes
       // (vertices)
       public function breadthFirstSearch($origin, $destination) {
              // mark all nodes as unvisited
              foreach ($this->graph as $vertex => $adj) {
                     $this->visited[$vertex] = false;
              }

              // create an empty queue
              $q = new SplQueue();

              // enqueue the origin vertex and mark as visited
              $q->enqueue($origin);
              $this->visited[$origin] = true;
```

```
		// this is used to track the path back from each node
		$path = array();
		$path[$origin] = new SplDoublyLinkedList();
		$path[$origin]->setIteratorMode(

	SplDoublyLinkedList::IT_MODE_FIFO|SplDoublyLinkedList::IT_MODE_KEEP
		);

		$path[$origin]->push($origin);

		$found = false;
		// while queue is not empty and destination not found
		while (!$q->isEmpty() && $q->bottom() != $destination) {
			$t = $q->dequeue();

			if (!empty($this->graph[$t])) {
				// for each adjacent neighbor
				foreach ($this->graph[$t] as $vertex) {
					if (!$this->visited[$vertex]) {
						// if not yet visited, enqueue vertex and
mark
						// as visited
						$q->enqueue($vertex);
						$this->visited[$vertex] = true;
						// add vertex to current path
						$path[$vertex] = clone $path[$t];
						$path[$vertex]->push($vertex);
					}
				}
			}
		}

		if (isset($path[$destination])) {
			echo "$origin to $destination in ",
			count($path[$destination]) - 1,
			" hops<br>";
					$sep = '';
					foreach ($path[$destination] as $vertex) {
					echo $sep, $vertex;
					$sep = '->';
					}
//					echo "<br>";
		}
		else {
			echo "No route from $origin to $destination<br>";
		}
	}
}

class Dijkstra
{
  protected $graph;

  public function __construct($graph) {
    $this->graph = $graph;
  }

  public function shortestPath($source, $target) {
    // array of best estimates of shortest path to each
    // vertex
    $d = array();
    // array of predecessors for each vertex
    $pi = array();
```

```
    // queue of all unoptimized vertices
    $Q = new SplPriorityQueue();

    foreach ($this->graph as $v => $adj) {
      $d[$v] = INF; // set initial distance to "infinity"
      $pi[$v] = null; // no known predecessors yet
      foreach ($adj as $w => $cost) {
        // use the edge cost as the priority
        $Q->insert($w, $cost);
      }
    }

    // initial distance at source is 0
    $d[$source] = 0;

    while (!$Q->isEmpty()) {
      // extract min cost
      $u = $Q->extract();
      if (!empty($this->graph[$u])) {
        // "relax" each adjacent vertex
        foreach ($this->graph[$u] as $v => $cost) {
          // alternate route length to adjacent neighbor
          $alt = $d[$u] + $cost;
          // if alternate route is shorter
          if ($alt < $d[$v]) {
            $d[$v] = $alt; // update minimum length to vertex
            $pi[$v] = $u;  // add neighbor to predecessors
                           //  for vertex
          }
        }
      }
    }

    // we can now find the shortest path using reverse
    // iteration
    $S = new SplStack(); // shortest path with a stack
    $u = $target;
    $dist = 0;
    // traverse from target to source
    while (isset($pi[$u]) && $pi[$u]) {
      $S->push($u);
      $dist += $this->graph[$u][$pi[$u]]; // add distance to predecessor
      $u = $pi[$u];
    }

    // stack will be empty if there is no route back
    if ($S->isEmpty()) {
      echo "No route from $source to $target";
    }
    else {
              // add the source node and print the path in reverse
              // (LIFO) order
              $S->push($source);
//            echo "$dist:";
              $sep = '';

              $con = mysqli_connect("localhost","root","","nrdata");

              // Check connection
              if (mysqli_connect_errno())
              {
                     echo "Failed to connect to MySQL: " . mysqli_connect_error();
              }
```

```
        $pathArray = array();
        $intCount = 0;

      foreach ($S as $v)
        {
             $tiplocQuery = "SELECT * FROM `nrtiploc` WHERE TIPLOCCodeID =
'".$v."'";
             $stmtTIPLOC = $con->prepare($tiplocQuery);
             $stmtTIPLOC->execute();
             $stmtTIPLOCresult = $stmtTIPLOC->get_result();
             while ($row = $stmtTIPLOCresult->fetch_assoc())
             {
                    $pathArray[++$intCount] = $v;
//                  echo $sep, $row['TIPLOCDescription'];
                    $sep = '->';
             }
      }
//      echo "<br>";
    }
  return $pathArray;
  }
}

if(empty($_GET['origin']) OR empty($_GET['destination']) OR
empty($_GET['departureTime']))
{
       echo "<HTML><HEAD>";
       echo "<link rel='stylesheet' type='text/css' href='stylesheet.css'>";
       echo "<TITLE>Multi-Train Information</TITLE></HEAD><BODY>";

//     echo "<div class='CSSTableGenerator'>";
       echo "<table border='1'>";
       echo "<tr>";
       echo "<td>Starting Station</td>";
       echo "<td>Ending Station</td>";
       echo "<td>Time From</td>";
       echo "<td>Time Until</td>";
       echo "<td>";
       echo "</tr>";
       echo "<FORM NAME='requestInfo' METHOD=''
ACTION='".$_SERVER['PHP_SELF']."'>";

       $stationQuery = "SELECT NRTIP.TIPLOCCodeID, NRTIP.TIPLOCDescription FROM
nrtiploc NRTIP STRAIGHT_JOIN nrscheduletrainsignals NRSIG ON (NRTIP.TIPLOCCodeID =
NRSIG.TIPLOCCode) GROUP BY NRTIP.TIPLOCCodeID, NRTIP.TIPLOCDescription ORDER BY
NRTIP.TIPLOCDescription";

       $stationStmt = $con->prepare($stationQuery);
       $stationStmt->execute();
       $stationResult = $stationStmt->get_result();

       echo "<td><select name='origin'>";
       while ($stationRow = $stationResult->fetch_assoc())
       {
             echo "<option
value='".$stationRow['TIPLOCCodeID']."'>".$stationRow['TIPLOCDescription']."</optio
n>";
       }
       echo "</select></td>";

       $stationQuery = "SELECT NRTIP.TIPLOCCodeID, NRTIP.TIPLOCDescription FROM
nrtiploc NRTIP STRAIGHT_JOIN nrscheduletrainsignals NRSIG ON (NRTIP.TIPLOCCodeID =
```

```
NRSIG.TIPLOCCode) GROUP BY NRTIP.TIPLOCCodeID, NRTIP.TIPLOCDescription ORDER BY
NRTIP.TIPLOCDescription";

	$stationStmt = $con->prepare($stationQuery);
	$stationStmt->execute();
	$stationResult = $stationStmt->get_result();

	echo "<td><select name='destination'>";
	while ($stationRow = $stationResult->fetch_assoc())
	{
		echo "<option
value='".$stationRow['TIPLOCCodeID']."'>".$stationRow['TIPLOCDescription']."</optio
n>";
	}
	echo "</select></td>";
	echo "<td><INPUT TYPE='text' NAME='departureTime'></td>";
	echo "<td><INPUT TYPE='Submit' NAME='Submit1' VALUE='Search Trains'></td>";
	echo "</tr>";
	echo "</FORM></table></div></body></html>";
}
else
{

	$graph = array();
	$dijkGraph = array();

	$thisQuery = "SELECT * FROM `nrfixedlinks` ORDER BY CRSOrigin,
CRSDestination";
	$thisStmt = $con->prepare($thisQuery);
	$thisStmt->execute();
	$thisResult = $thisStmt->get_result();
	while ($row = $thisResult->fetch_assoc())
	{
		$origin = $row['CRSOrigin'];
		$destination = $row['CRSDestination'];
		$length = $row['linkMinutes'];
		if(array_key_exists($origin, $graph)) {
			if(is_array($graph[$origin])) {
				$graph[$origin][] = $destination;
			}
			else {
				$graph[$origin] = array($graph[$origin],
$destination);
			}
		}
		else
		{
			$graph[$origin] = array($destination);
		}

		if(array_key_exists($origin, $dijkGraph)) {
			if(is_array($dijkGraph[$origin])) {
				$dijkGraph[$origin][$destination] = $length;
			}
			else {
				$dijkGraph[$origin][$destination] =
array($graph[$origin][$destination], $length);
			}
		}
		else
		{
			$dijkGraph[$origin][$destination] = $length;
		}
```

```
	}

	$g = new Graph($graph);

	$h = new Dijkstra($dijkGraph);

	$origin = $_GET['origin'];
	$destination = $_GET['destination'];
	$departureTime = $_GET['departureTime'];

	$pathingArray = $h->shortestPath($origin, $destination);

	$numStations = count($pathingArray);
	$countStation = $numStations;
	$currentStation = 1;
	$numResults = 0;
	$trainRoute = array();

	 // This next section calculates how many separate transport links are
needed to get from Point A to Point B.

	do
	{
		do
		{
			if ($currentStation == $countStation)
			{
				$pathingQuery = "SELECT * FROM nrfixedlinks WHERE
CRSOrigin = '".$pathingArray[$currentStation]."' AND CRSDestination =
'".$pathingArray[($countStation + 1)]."'";
				$pathingStmt = $con->prepare($pathingQuery);
				$pathingStmt->execute();
				$pathingResult = $pathingStmt->get_result();
				$numResults = mysqli_num_rows($pathingResult);
				if ($numResults > 0)
				{
					$nextLeg = count($trainRoute);
					$trainRoute[$nextLeg] = array("origin" =>
$pathingArray[$currentStation], "destination" => $pathingArray[($countStation + 1)],
"type" => "WALK");
				}
			}
			else
			{
				$pathingQuery = "SELECT NRSCH.scheduleTrainID,
NRSCH.signallingID, NRSCH.daysRun, NRSCH.scheduleStartDate, NRSCH.scheduleEndDate,
NRORG.TIPLOCCode AS originTIPLOC, NRORG.publicDeparture, NRSIG.TIPLOCCode AS
destinationTIPLOC, NRSIG.publicArrival FROM nrscheduletrains NRSCH INNER JOIN
nrscheduletrainsignals NRSIG ON NRSCH.scheduletrainID = NRSIG.signalTrainID INNER
JOIN (SELECT * FROM nrscheduletrains INNER JOIN nrscheduletrainsignals ON
nrscheduletrains.scheduletrainID = nrscheduletrainsignals.signalTrainID WHERE
nrscheduletrainsignals.locationType <> 'LT' AND nrscheduletrainsignals.TIPLOCCode =
'".$pathingArray[$currentStation]."' ORDER BY nrscheduletrains.scheduleTrainID ASC,
nrscheduletrainsignals.signalPointNo ASC) AS NRORG ON NRSCH.scheduleTrainID =
NRORG.scheduleTrainID AND NRSCH.scheduleStartDate = NRORG.scheduleStartDate AND
NRSCH.scheduleEndDate = NRORG.scheduleEndDate WHERE NRSIG.locationType <> 'LO' AND
NRSIG.TIPLOCCode = '".$pathingArray[$countStation]."' AND NRSCH.scheduleStartDate
<= '$today' AND NRSCH.scheduleEndDate >= '$today' AND NRSIG.signalPointNo >
NRORG.signalPointNo AND NRSIG.internalArrival <> 'pass' AND (NRSCH.trainCategory =
'OO' OR NRSCH.trainCategory = 'XX') ORDER BY NRORG.internalDeparture ASC,
NRSCH.scheduleTrainID ASC, NRSCH.scheduleStartDate ASC";
```

```
                        $pathingStmt = $con->prepare($pathingQuery);
                        $pathingStmt->execute();
                        $pathingResult = $pathingStmt->get_result();
                        $countStation--;
                        $numResults = mysqli_num_rows($pathingResult);
                        if ($numResults > 0)
                        {
                               $nextLeg = count($trainRoute);
                               $trainRoute[$nextLeg] = array("origin" =>
$pathingArray[$currentStation], "destination" => $pathingArray[($countStation + 1)],
"type" => "TRAIN");
                        }
                }
        } while ($numResults == 0);
        $currentStation = $countStation + 1;
        $countStation = $numStations;
} while ($currentStation != $numStations);

echo "<HTML><HEAD>";
echo "<link rel='stylesheet' type='text/css' href='stylesheet.css'>";
echo "<TITLE>Route Finder for Multi-Leg Trips</TITLE></HEAD><BODY>";

echo "<center>Route from $origin to $destination</center><p>";
echo "<div class='CSSTableGenerator'>";
echo "<table border='1'>";
echo "<tr>";
echo "<td>Mode (Train ID)</td>";
echo "<td>Origin</td>";
echo "<td>Time</td>";
echo "<td>Destination</td>";
echo "<td>Time</td>";
echo "</tr>";
echo "</div>";

foreach ($trainRoute as $leg => $details)
{
        if ($details['type'] == "TRAIN")
        {
                $nextLegQuery = "SELECT NRSCH.scheduleTrainID,
NRSCH.signallingID, NRSCH.daysRun, NRSCH.scheduleStartDate, NRSCH.scheduleEndDate,
NRORG.TIPLOCCode AS originTIPLOC, NRORG.publicDeparture, NRSIG.TIPLOCCode AS
destinationTIPLOC, NRSIG.publicArrival FROM nrscheduletrains NRSCH INNER JOIN
nrscheduletrainsignals NRSIG ON (NRSCH.scheduletrainID = NRSIG.signalTrainID AND
NRSCH.scheduleStartDate = NRSIG.signalStartDate AND NRSCH.scheduleEndDate =
NRSIG.signalEndDate) INNER JOIN (SELECT * FROM nrscheduletrains INNER JOIN
nrscheduletrainsignals ON nrscheduletrains.scheduletrainID =
nrscheduletrainsignals.signalTrainID WHERE nrscheduletrainsignals.locationType <>
'LT' AND nrscheduletrainsignals.TIPLOCCode = '".$details['origin']."' AND
nrscheduletrains.scheduleStartDate = nrscheduletrainsignals.signalStartDate AND
nrscheduletrains.scheduleEndDate = nrscheduletrainsignals.signalEndDate ORDER BY
nrscheduletrains.scheduleTrainID ASC, nrscheduletrainsignals.signalPointNo ASC) AS
NRORG ON NRSCH.scheduleTrainID = NRORG.scheduleTrainID AND NRSCH.scheduleStartDate
= NRORG.scheduleStartDate AND NRSCH.scheduleEndDate = NRORG.scheduleEndDate WHERE
NRSIG.locationType <> 'LO' AND NRSIG.TIPLOCCode = '".$details['destination']."' AND
NRORG.internalDeparture >= '$departureTime' AND NRSCH.scheduleStartDate <= '2014-
08-05' AND NRSCH.scheduleEndDate >= '2014-08-05' AND NRSIG.signalPointNo >
NRORG.signalPointNo AND NRSIG.internalArrival <> 'pass' AND NRSCH.daysRun =
'1111100' AND (NRSCH.trainCategory = 'OO' OR NRSCH.trainCategory = 'XX') ORDER BY
NRORG.internalDeparture ASC, NRSCH.scheduleTrainID ASC, NRSCH.scheduleStartDate ASC
LIMIT 3";
                $nextLegStmt = $con->prepare($nextLegQuery);
                $nextLegStmt->execute();
                $nextLegResult = $nextLegStmt->get_result();
```

```
                    $numResults = mysqli_num_rows($nextLegResult);
                    if ($numResults > 0)
                    {
                           while ($row = $nextLegResult->fetch_assoc())
                           {
                                  echo "<tr>";
                                  echo "<td>Train:
".$row["scheduleTrainID"]."</td>";
                                  $stationQuery = "SELECT * FROM nrtiploc WHERE
TIPLOCCodeID = '".$row['originTIPLOC']."'";
                                  $stationStmt = $con->prepare($stationQuery);
                                  $stationStmt->execute();
                                  $stationResult = $stationStmt->get_result();
                                  while ($stationRow = $stationResult-
>fetch_assoc()) {
                                         echo
"<td>".ucwords(strtolower($stationRow['TIPLOCDescription'])))."</td>";
                                  }
                                  echo "<td>".$row["publicDeparture"]."</td>";
                                  $stationQuery = "SELECT * FROM nrtiploc WHERE
TIPLOCCodeID = '".$row['destinationTIPLOC']."'";
                                  $stationStmt = $con->prepare($stationQuery);
                                  $stationStmt->execute();
                                  $stationResult = $stationStmt->get_result();
                                  while ($stationRow = $stationResult-
>fetch_assoc()) {
                                         echo
"<td>".ucwords(strtolower($stationRow['TIPLOCDescription'])))."</td>";
                                  }
                                  echo "<td>".$row["publicArrival"]."</td>";
                                  $departureTime = $row["publicArrival"];
                                  echo "</tr>";
                           }
                    }
             }
             else
             {
                    $nextLegQuery = "SELECT * FROM nrfixedlinks WHERE CRSOrigin =
'".$details['origin']."' AND CRSDestination = '".$details['destination']."'";
                    $nextLegStmt = $con->prepare($nextLegQuery);
                    $nextLegStmt->execute();
                    $nextLegResult = $nextLegStmt->get_result();
                    $numResults = mysqli_num_rows($nextLegResult);
                    if ($numResults > 0)
                    {
                           while ($row = $nextLegResult->fetch_assoc())
                           {
                                  echo "<tr class='CSSWalkingGenerator'>";
                                  echo "<td class='CSSWalkingGenerator'>Walk</td>";
                                  $stationQuery = "SELECT * FROM nrtiploc WHERE
TIPLOCCodeID = '".$row['CRSOrigin']."'";
                                  $stationStmt = $con->prepare($stationQuery);
                                  $stationStmt->execute();
                                  $stationResult = $stationStmt->get_result();
                                  while ($stationRow = $stationResult-
>fetch_assoc()) {
                                         echo "<td
class='CSSWalkingGenerator'>".ucwords(strtolower($stationRow["TIPLOCDescription"]))
."</td>";
                                  }
                                  echo "<td
class='CSSWalkingGenerator'>$departureTime</td>";
```

```
                                    $stationQuery = "SELECT * FROM nrtiploc WHERE
TIPLOCCodeID = '".$row['CRSDestination']."'";
                                    $stationStmt = $con->prepare($stationQuery);
                                    $stationStmt->execute();
                                    $stationResult = $stationStmt->get_result();
                                    while ($stationRow = $stationResult-
>fetch_assoc()) {
                                            echo "<td
class='CSSWalkingGenerator'>".ucwords(strtolower($stationRow["TIPLOCDescription"]))
."</td>";
                                    }
                                    $transitTime = DateTime::createFromFormat('Hi',
$departureTime);
                                    $transitTime->add(new
DateInterval('PT'.($row["linkMinutes"]).'M'));
                                    echo "<td
class='CSSWalkingGenerator'>".$transitTime->format('Hi')."</td>";
                                    echo "</tr>";
                                    $departureTime = $transitTime->format('Hi');
                                }
                            }
                        }
                }
                echo "</tr></table></div></body></html>";
}
?>
```

Page: NRLiveData_SQLParser.php

```
<?php

/*==================================
FILE: NRLiveData_SQLParser.php

PURPOSE: To access the RealTime Trains database, extract the relevant data, and
insert it into our database
         for analysis and display. There is no front end for this, as this would be
done by an administrator.

CREDITS: All code by Matt Griffin.
         Special thanks to Tom Curtis for the use of his database at RealTime
Trains

==================================*/

//Extend PHP's maximum script execution time in order to process the required data
//in this case, allow execution to continue until the end of script is reached or
an error is thrown
ini_set('max_execution_time', 0);
ini_set('max_input_time', 0);

//connect to MySQL database
$con = mysqli_connect("localhost","root","","nrdata");

// Check connection
if (mysqli_connect_errno())
{
       echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

//select the start and end dates for data retrieval
$date = '2014-08-10';
$end_date = '2014-08-24';

//while the end date has not been reached
while (strtotime($date) <= strtotime($end_date))
{
        //set up the insertion query
       $query = "INSERT INTO NRHistoricData (trainID, dateRun, trainStop,
trainTIPLOC, trainTIPLOCDescription, bookedArrival, bookedDeparture, RTarrival,
RTdeparture, RTlateness, trainStopStatus, trainCancelReason) VALUES ";

        //get the list of all trains from the schedule table that should have run
on this date
       $trainIDQuery = "SELECT * FROM `nrscheduletrains` WHERE `scheduleStartDate`
<= '$date' AND `scheduleEndDate` >= '$date' AND (`trainCategory` = 'OU' OR
`trainCategory` = 'OO' OR `trainCategory` = 'XC' OR `trainCategory` = 'XX')";
       $trainIDresult = $con->query($trainIDQuery);

       $arrTrainServices = array();

        //extract all the schedule IDs from the schedule table and insert them into
an array
       while($row = $trainIDresult->fetch_assoc()){
              $arrDaysRun = str_split($row['daysRun']);
              $weekDay = date('N', strtotime($date));
              if(($arrDaysRun[$weekDay - 1]) == "1")
              {
                     $arrTrainServices[] = $row['scheduleTrainID'];
              }
```

```
             else echo "";
      }

        //calculate the number of trains using the size of the array, then loop
until that number is reached
      for($i = 0, $size = count($arrTrainServices); $i < $size; ++$i)
      {
            echo "Parsing record $i...";
            echo "\n";
            $dateSearch = explode('-',$date);

            //assemble the correct URL to access the RealTime Trains API. Each
train has its own URL formed by the ID and the date it ran
            //data is only available from two months before the current date
            $service_url =
'https://secure.realtimetrains.co.uk/api/json/service/'.$arrTrainServices[$i].'/';
            $service_url .= $dateSearch[0];
            $service_url .= '/';
            $service_url .= $dateSearch[1];
            $service_url .= '/';
            $service_url .= $dateSearch[2];

            //this initialises a cURL request for accessing the API. cURL (Client
URL) is a way of making remote requests for data using PHP,
            //and the RealTime Trains developer recommends using this to connect to
his data set.
            $curl = curl_init($service_url);
            curl_setopt($curl, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
            //this can be changed by requesting a new account from RealTime Trains
            curl_setopt($curl, CURLOPT_USERPWD,
"rttapi_griffin:99929e6ade037853930b91c220d231cd0232a3f9");
            curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
            curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, false);
            $curl_response = curl_exec($curl);
            //if cURL does not give a response, then close the request and report
back to the user
            if ($curl_response === false) {
                    $info = curl_getinfo($curl);
                    echo curl_error($curl)."<p>";
                    curl_close($curl);
                    die('error occured during curl exec. Additional info: ' .
var_export($info));
            }
            curl_close($curl);
            if (isset($decoded->response->status) && $decoded->response->status ==
'ERROR') {
                    die('error occured: ' . $decoded->response->errormessage);
                    }

            $countStations = 0;

            //the URL called returns a JSON string. We need to decode this before
we can process it further
            $json_result = json_decode($curl_response, true);

            if(!empty($json_result['error'])) echo "";
            //if it turns out that we have called a freight train or other non-
passenger train, do not process the service
            elseif($json_result['isPassenger'] == FALSE) echo "";
            else
            {
                    //each train is made up of multiple stops, we need to process
them one by one
```

```
                    foreach ($json_result['locations'] as $value) {
                            $countStations++;
                            //building the insertion query by adding train detail,
then information about the stop
                            $query .= "('".$json_result['serviceUid']."', ";
                            $query .= "'".$json_result['runDate']."', ";
                            $query .= $countStations.", ";
                            $query .= "'".$value['tiploc']."', ";
                            //cleaning the stop name in case there are special
characters
                            $query .= "'".addslashes($value['description'])."', ";
                            //check to see if there are any blank values in the
arrival/departure times
                            if (empty($value['gbttBookedArrival'])) $query .= "NULL,
";
                            else $query .= "'".$value['gbttBookedArrival']."', ";
                            if (empty($value['gbttBookedDeparture'])) $query .=
"NULL, ";
                            else $query .= "'".$value['gbttBookedDeparture']."', ";
                            if ((empty($value['realtimeArrivalActual'])) or
($value['displayAs'] == "CANCELLED_CALL")) $query .= "NULL, ";
                            else {
                                    if ($value['realtimeArrivalActual'] = "true")
$query .= "'".$value['realtimeArrival']."', ";
                                    //it is possible that there are no timing
points at the stop, or that there is a failure in the reporting system
                                    //in this case, we are returning "No Report" to
indicate that the time was not recorded
                                    else $query .= "'No Report', ";
                            }
                            if (empty($value['realtimeDepartureActual'])) $query .=
"NULL, ";
                            else {
                                    if ($value['realtimeDepartureActual'] = "true")
$query .= "'".$value['realtimeDeparture']."', ";
                                    else $query .= "'No Report', ";
                            }
                            if (empty($value['realtimeGbttDepartureLateness']))
$query .= "NULL, ";
                            else $query .=
"'".$value['realtimeGbttDepartureLateness']."', ";
                            if (empty($value['displayAs'])) $query .= "NULL, ";
                            else $query .= "'".$value['displayAs']."', ";
                            if (empty($value['cancelReasonShortText'])) $query .=
"NULL), ";
                            else $query .=
"'".addslashes($value['cancelReasonLongText'])."'), ";
                    }
            }
       }
        //prepare the query by removing the training comma, then bind and execute
the query
        $query = rtrim($query, " ,");
       $stmt = $con->prepare($query);
       $stmt->execute();
       $result = $stmt->get_result();
        //then increment the date
        $date = date ("Y-m-d", strtotime("+1 day", strtotime($date)));
}
?>
```

Page: NRMultiServiceAtStation.php

```
<?php

/*==================================
FILE: NRMultiServiceInfoAtStation.php

PURPOSE: To display all trains that stop at the selected station during the
selected time period.
         This shows information on the trains, their origin and destination,
arrival and departure
         times, average arrival and departure, plus additional summary information.
         This could be of use to railway staff and platform attendants.

CREDITS: All work by Matt Griffin

==================================*/

//Extend PHP's maximum script execution time in order to process the required data
ini_set('max_execution_time', 6000);

//connect to the database
$con = mysqli_connect("localhost","root","","nrdata");

// Check connection
if (mysqli_connect_errno())
{
       echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

//use today's date to find the current schedule and declare an array of dates
$today = date("Y-m-d");
$dateArray = array();

echo "<HTML><HEAD>";
echo "<link rel='stylesheet' type='text/css' href='stylesheet.css'>";
echo "<TITLE>Multi-Train Information</TITLE></HEAD><BODY>";

//show selection table if not already displayed or information not selected
if(empty($_GET['station']) OR empty($_GET['startTime']) OR empty($_GET['endTime']))
{
        echo "<CENTER><b>Station Traffic Selection - Please selection station and
time span to view</b></center><p>";
       echo "<div class='CSSTableGenerator'>";
       echo "<table border='1'>";
       echo "<tr>";
       echo "<td>Station</td>";
       echo "<td>Time From</td>";
       echo "<td>Time Until</td>";
       echo "<td>Scheduled Days</td>";
       echo "<td></td>";
       echo "</tr>";
       echo "<FORM NAME='requestInfo' METHOD=''
ACTION='".$_SERVER['PHP_SELF']."'>";

        //run query to pick up all station names for selection boxes
        $stationQuery = "SELECT NRTIP.TIPLOCCodeID, NRTIP.TIPLOCDescription FROM
nrtiploc NRTIP STRAIGHT_JOIN nrscheduletrainsignals NRSIG ON (NRTIP.TIPLOCCodeID =
NRSIG.TIPLOCCode) GROUP BY NRTIP.TIPLOCCodeID, NRTIP.TIPLOCDescription ORDER BY
NRTIP.TIPLOCDescription";

       $stationStmt = $con->prepare($stationQuery);
```

```
	$stationStmt->execute();
	$stationResult = $stationStmt->get_result();

	 //display selection table, including boxes for station and start/end times
	 echo "<td><select name='station'>";
	while ($stationRow = $stationResult->fetch_assoc())
	{
		echo "<option
value='".$stationRow['TIPLOCCodeID']."'>".$stationRow['TIPLOCDescription']."</optio
n>";
	}
	echo "</select></td>";

	echo "<td><INPUT TYPE='text' NAME='startTime'></td>";
	echo "<td><INPUT TYPE='text' NAME='endTime'></td>";
	echo "<td><SELECT NAME='daysRun'><option
value='1111100'>Weekdays</option><option value='0000010'>Saturdays</option><option
value='0000001'>Sundays</option><option value=''>All Days</option></select></td>";
	echo "<td><INPUT TYPE='Submit' NAME='Submit1' VALUE='Search Trains'></td>";
	echo "</tr>";
	echo "</FORM></table></div></body></html>";
}

else
{
	 //allocate posted variables to local variables
	 $stationTIPLOC = $_GET['station'];
	$commencementTime = $_GET['startTime'];
	$terminationTime = $_GET['endTime'];
	if((isset($_GET['daysRun'])) AND (!empty($_GET['daysRun']))) $daysRun =
$_GET['daysRun'];

	 //these will be the first and last dates used for data selection from the
database
	$firstDate = new DateTime('2014-05-19');
	$lastDate = new DateTime('2014-08-08');

	 //add all dates between first and last to the date array declared earlier
	while ($firstDate <= $lastDate)
	{
		array_push($dateArray, $firstDate->format('Y-m-d'));
		$firstDate->add(new DateInterval('P1D'));
	}

	 //run query to return station details
		$stationQuery = "SELECT * FROM nrtiploc WHERE TIPLOCCodeID =
'".$stationTIPLOC."'";

	$stationStmt = $con->prepare($stationQuery);
	$stationStmt->execute();
	$stationResult = $stationStmt->get_result();

	 //convert station name from uppercase to capitalise first letter
	 while ($stationRow = $stationResult->fetch_assoc()) {
			$stationDescription =
ucwords(strtolower($stationRow['TIPLOCDescription']));
	 }

	 //run query to return all trains travelling through station between start
and end time
	 $detailQuery = "SELECT NRSCH.scheduleTrainID, NRSCH.signallingID,
NRSCH.trainOperator, NRSCH.daysRun, NRSCH.scheduleStartDate, NRSCH.scheduleEndDate,
NRORG.locationType, NRORG.TIPLOCCode AS originTIPLOC, NRORG.publicArrival,
```

```
NRORG.publicDeparture FROM nrscheduletrains NRSCH INNER JOIN (SELECT * FROM
nrscheduletrains INNER JOIN nrscheduletrainsignals ON
nrscheduletrains.scheduletrainID = nrscheduletrainsignals.signalTrainID AND
nrscheduletrains.scheduleStartDate = nrscheduletrainsignals.signalStartDate AND
nrscheduletrains.scheduleEndDate = nrscheduletrainsignals.signalEndDate WHERE
nrscheduletrainsignals.TIPLOCCode = '$stationTIPLOC' ORDER BY
nrscheduletrains.scheduleTrainID ASC, nrscheduletrainsignals.signalPointNo ASC) AS
NRORG ON NRSCH.scheduleTrainID = NRORG.scheduleTrainID AND NRSCH.scheduleStartDate
= NRORG.scheduleStartDate AND NRSCH.scheduleEndDate = NRORG.scheduleEndDate WHERE
GREATEST(COALESCE(NRORG.internalArrival,0),COALESCE(NRORG.internalDeparture,0)) >=
'$commencementTime' AND
GREATEST(COALESCE(NRORG.internalArrival,0),COALESCE(NRORG.internalDeparture,0)) <=
'$terminationTime' AND NRSCH.scheduleStartDate <= '$today' AND
NRSCH.scheduleEndDate >= '$today' AND (NRSCH.trainCategory = 'OO' OR
NRSCH.trainCategory = 'XX') ";
        //this line will limit query to weekdays/weekends/all days
        if(isset($daysRun)) $detailQuery .= "AND NRSCH.daysRun = '$daysRun' ";
        //the COALESCE function in MySQL allows us to return either the
arrival/departure time or 0, depending on whether it exists
        $detailQuery .= "ORDER BY GREATEST(COALESCE(NRORG.publicArrival,0),
COALESCE(NRORG.publicDeparture,0)) ASC, NRSCH.scheduleTrainID ASC,
NRSCH.scheduleStartDate ASC";
        $detailStmt = $con->prepare($detailQuery);
       $detailStmt->execute();
       $detailResult = $detailStmt->get_result();
        $numTrains = mysqli_num_rows($detailResult);

        //prepare the header display information
        echo "<CENTER><b>Daily train performance - trains calling at
$stationDescription between $commencementTime and $terminationTime";
       if(isset($daysRun))
        {
            if ($daysRun == "1111100")
            {
                $strDays = " (weekday services only)";
            }
            else if ($daysRun == "0000010")
            {
                $strDays = " (Saturday services only)";
            }
            else if ($daysRun == "0000001")
            {
                $strDays = " (Sunday services only)";
            }
            else
            {
                $strDays = "";
            }
            echo $strDays." - $numTrains services<br>";
        }
        else
        {
            echo " - $numTrains services<br>";
        }
        echo "</b></center><p>";

        //build table header
        echo "<div class='CSSTableGenerator'>";
       echo "<table border='1'>";
       echo "<tr>";
       echo "<td>Train ID</td>";
       echo "<td>Signalling ID</td>";
       echo "<td>Train Operator</td>";
```

```
        echo "<td>Days Run</td>";
        echo "<td>Origin</td>";
        echo "<td>Destination</td>";
        echo "<td>Arrival Time</td>";
        echo "<td>Departure Time</td>";
        echo "<td>Average Arrival Time</td>";
        echo "<td>Average Departure Time</td>";
        echo "<td>Trains Run</td>";
        echo "<td>On Time</td>";
        echo "<td>% On Time</td>";
        echo "<td>% within 5m</td>";
        echo "</tr>";

         //loop through each result from query and output information for each train
        while ($row = $detailResult->fetch_assoc()) {
                echo "<tr>";
               $strDays = "";
                //return information on origin and destination
                $originQuery = "SELECT * from nrscheduletrainsignals NRSCH
STRAIGHT_JOIN nrtiploc NRTIP ON (NRSCH.TIPLOCCode = NRTIP.TIPLOCCodeID) WHERE
NRSCH.signalTrainID = '".$row["scheduleTrainID"]."' AND NRSCH.locationType = 'LO'";
               $originStmt = $con->prepare($originQuery);
               $originStmt->execute();
               $originResult = $originStmt->get_result();

               $destinationQuery = "SELECT * from nrscheduletrainsignals NRSCH
STRAIGHT_JOIN nrtiploc NRTIP ON (NRSCH.TIPLOCCode = NRTIP.TIPLOCCodeID) WHERE
NRSCH.signalTrainID = '".$row["scheduleTrainID"]."' AND NRSCH.locationType = 'LT'";
               $destinationStmt = $con->prepare($destinationQuery);
               $destinationStmt->execute();
               $destinationResult = $destinationStmt->get_result();

               while ($originRow = $originResult->fetch_assoc()) {
                      while ($destinationRow = $destinationResult->fetch_assoc()) {
                              echo "<td>".$row["scheduleTrainID"]."</td>";
                              echo "<td><a
href='NRSingleTrainInfo.php?scheduleTrainID=".$row["scheduleTrainID"]."&origin=".$o
riginRow['TIPLOCCode']."&destination=".$destinationRow['TIPLOCCode']."'>".$row["sig
nallingID"]."</a></td>";
                              echo "<td>".$row["trainOperator"]."</td>";
                              $strDaysRun = $row["daysRun"];
                              //return info on schedules in a more user-friendly
format
                              if ($strDaysRun == "1111100") $strDays = "Mon-Fri";
                              else
                              {
                                      if (substr($strDaysRun,0,1) == "1") $strDays .=
"Mon/";
                                      if (substr($strDaysRun,1,1) == "1") $strDays .=
"Tue/";
                                      if (substr($strDaysRun,2,1) == "1") $strDays .=
"Wed/";
                                      if (substr($strDaysRun,3,1) == "1") $strDays .=
"Thu/";
                                      if (substr($strDaysRun,4,1) == "1") $strDays .=
"Fri/";
                                      if (substr($strDaysRun,5,1) == "1") $strDays .=
"Sat/";
                                      if (substr($strDaysRun,6,1) == "1") $strDays .=
"Sun/";
                                      $strDays = rtrim($strDays, "/");
                              }
                              echo "<td>".$strDays."</td>";
```

```
                            echo "<td>".$originRow["TIPLOCDescription"]."</td>";
                            echo
"<td>".$destinationRow["TIPLOCDescription"]."</td>";
                            //convert the times to more readable formats - hours
and minutes
                            if ($row["publicArrival"] == NULL) echo "<td></td>";
                            else echo
"<td>".date('H:i',strtotime($row["publicArrival"])."</td>";
                            if ($row["publicDeparture"] == NULL) echo "<td></td>";
                            else echo
"<td>".date('H:i',strtotime($row["publicDeparture"])."</td>";
                //return the average arrival time for each train, to nearest second
                //in order to calculate average, the time needs to be converted
into milliseconds and then back again for calculation
                //this would be more useful with more accurate source data
                $averageQuery = "SELECT
DATE_FORMAT(FROM_UNIXTIME(AVG(UNIX_TIMESTAMP(STR_TO_DATE(RTarrival,'%H%i')))),'%H:%
i:%s') AS 'Actual Arrival' FROM nrhistoricdata NRHIS WHERE NRHIS.trainID =
'".$row["scheduleTrainID"]."' AND NRHIS.trainTIPLOC = '".$stationTIPLOC."' AND
NRHIS.RTarrival IS NOT NULL";
             $averageStmt = $con->prepare($averageQuery);
             $averageStmt->execute();
             $averageResult = $averageStmt->get_result();
                while ($averageRow = $averageResult->fetch_assoc())
                {
                    echo "<td>".$averageRow['Actual Arrival']."</td>";
                }

                //return the average departure time for each train, to nearest
second
                //as above, this would be more useful with accurate source data
                $averageDepartureQuery = "SELECT
DATE_FORMAT(FROM_UNIXTIME(AVG(UNIX_TIMESTAMP(STR_TO_DATE(RTdeparture,'%H%i')))),'%H
:%i:%s') AS 'Actual Departure' FROM nrhistoricdata NRHIS WHERE NRHIS.trainID =
'".$row["scheduleTrainID"]."' AND NRHIS.trainTIPLOC = '".$stationTIPLOC."' AND
NRHIS.RTdeparture IS NOT NULL";
             $averageDepartureStmt = $con->prepare($averageDepartureQuery);
             $averageDepartureStmt->execute();
             $averageDepartureResult = $averageDepartureStmt->get_result();
                while ($averageDepartureRow = $averageDepartureResult-
>fetch_assoc())
                {
                    echo "<td>".$averageDepartureRow['Actual Departure']."</td>";
                }

                //return number of times that this train ran
            if ($row["locationType"] == "LT") $calcQuery = "SELECT scheduleTrainID,
STR_TO_DATE(bookedArrival, '%H%i') AS 'Booked Arrival', COUNT(scheduleTrainID) AS
'Times Run', SUM(IF(RTArrival IS NULL, 0,IF(STR_TO_DATE(RTArrival,'%H%i') <=
STR_TO_DATE(bookedArrival, '%H%i'), 1, 0))) AS 'On Time', SUM(IF(RTArrival IS NULL,
0,IF(TIME_TO_SEC(TIMEDIFF(STR_TO_DATE(RTArrival,'%H%i'),STR_TO_DATE(bookedArrival,
'%H%i')))/60 >= 5, 0, 1))) AS 'Times > 5m Late' FROM `nrscheduletrains` INNER JOIN
`nrscheduletrainsignals` ON (`nrscheduletrains`.scheduleTrainID =
`nrscheduletrainsignals`.signalTrainID) INNER JOIN `nrhistoricdata` ON
(`nrscheduletrains`.scheduleTrainID = `nrhistoricdata`.trainID AND
`nrscheduletrainsignals`.TIPLOCCode = `nrhistoricdata`.trainTIPLOC AND
`nrscheduletrainsignals`.TIPLOCCode = '$stationTIPLOC' AND
`nrhistoricdata`.trainTIPLOC = '$stationTIPLOC') WHERE
`nrscheduletrains`.scheduleTrainID = '".$row['scheduleTrainID']."' AND
`nrscheduletrainsignals`.publicArrival = `nrhistoricdata`.bookedArrival AND dateRun
>= '".$dateArray[0]."' and dateRun <= '".$lastDate->format('Y-m-d')."' AND
`nrscheduletrains`.scheduleStartDate = '".$row["scheduleStartDate"]."' AND
```

```
SUBSTRING(`nrscheduletrains`.daysRun,WEEKDAY(dateRun)+1,1) = '1' GROUP BY
scheduleTrainID, bookedArrival ORDER BY bookedArrival ASC, RTArrival ASC";
            else $calcQuery = "SELECT scheduleTrainID, STR_TO_DATE(bookedDeparture,
'%H%i') AS 'Booked Departure', COUNT(scheduleTrainID) AS 'Times Run',
SUM(IF(RTDeparture IS NULL, 0,IF(STR_TO_DATE(RTDeparture,'%H%i') <=
STR_TO_DATE(bookedDeparture, '%H%i'), 1, 0))) AS 'On Time', SUM(IF(RTDeparture IS
NULL,
0,IF(TIME_TO_SEC(TIMEDIFF(STR_TO_DATE(RTDeparture,'%H%i'),STR_TO_DATE(bookedDepartu
re, '%H%i')))/60 >= 5, 0, 1))) AS 'Times > 5m Late' FROM `nrscheduletrains` INNER
JOIN `nrscheduletrainsignals` ON (`nrscheduletrains`.scheduleTrainID =
`nrscheduletrainsignals`.signalTrainID AND `nrscheduletrains`.scheduleStartDate =
`nrscheduletrainsignals`.signalStartDate AND `nrscheduletrains`.scheduleEndDate =
`nrscheduletrainsignals`.signalEndDate) INNER JOIN `nrhistoricdata` ON
(`nrscheduletrains`.scheduleTrainID = `nrhistoricdata`.trainID AND
`nrscheduletrainsignals`.TIPLOCCode = `nrhistoricdata`.trainTIPLOC AND
`nrscheduletrainsignals`.TIPLOCCode = '$stationTIPLOC' AND
`nrhistoricdata`.trainTIPLOC = '$stationTIPLOC') WHERE
`nrscheduletrains`.scheduleTrainID = '".$row['scheduleTrainID']."' AND
`nrscheduletrainsignals`.publicDeparture = `nrhistoricdata`.bookedDeparture AND
dateRun >= `nrscheduletrains`.scheduleStartDate and dateRun <=
`nrscheduletrains`.scheduleEndDate AND dateRun >= '".$dateArray[0]."' and dateRun
<= '".$lastDate->format('Y-m-d')."' AND
SUBSTRING(`nrscheduletrains`.daysRun,WEEKDAY(dateRun)+1,1) = '1' GROUP BY
scheduleTrainID, bookedDeparture ORDER BY bookedDeparture ASC, RTDeparture ASC";
               $calcStmt = $con->prepare($calcQuery);
            $calcStmt->execute();
            $calcResult = $calcStmt->get_result();

               //return number of times that this train did not run due to
cancellation
            $noServiceQuery = "SELECT trainID, COUNT(trainID) AS 'No Service' FROM
nrhistoricdata WHERE trainID = '".$row['scheduleTrainID']."' AND trainTIPLOC =
'$stationTIPLOC' AND dateRun >= '".$dateArray[0]."' AND dateRun <= '".$lastDate-
>format('Y-m-d')."' AND trainCancelReason LIKE '%%' GROUP BY trainID ORDER BY
bookedArrival ASC, RTarrival ASC";
               $noServiceStmt = $con->prepare($noServiceQuery);
            $noServiceStmt->execute();
            $noServiceResult = $noServiceStmt->get_result();

               //display percentage of trains that arrived (a) on time, and (b)
within 5 minutes of schedule
               while ($calcRow = $calcResult->fetch_assoc()) {
                   $rowCountNoService = $noServiceResult->num_rows;
                       //this checks to see if there are any results from the
cancelled trains query
                       //done to ensure that there are no issues with the
fetch_assoc call from the result set
                   if ($rowCountNoService == 0)
                   {
                          $numTimesRun = $calcRow['Times Run'];
                          echo "<td>".$numTimesRun."</td>";
                          echo "<td>".$calcRow['On Time']."</td>";
                          echo "<td>".round(($calcRow['On Time'] / $numTimesRun) *
100,1)."%</td>";
                          echo "<td>".round(($calcRow['Times > 5m Late'] /
$numTimesRun) * 100,1)."%</td>";
                   }
                   else
                   {
                          while ($noServiceRow = $noServiceResult->fetch_assoc())
{
                                 $numTimesRun = $calcRow['Times Run'] -
$noServiceRow['No Service'];
```

```
                                    echo "<td>".$numTimesRun."</td>";
                                    echo "<td>".$calcRow['On Time']."</td>";
                                    echo "<td>".round(($calcRow['On Time'] /
$numTimesRun) * 100,1)."%</td>";
                                    echo "<td>".round(($calcRow['Times > 5m Late'] /
$numTimesRun) * 100,1)."%</td>";
                                }
                            }
                        }
                        echo "</tr>";
                    }
                }
            }
            echo "</table></div></body></html>";
}
?>
```

Page: NRMultiTrainInfo.php

```
<?php

/*==================================
FILE: NRMultiTrainInfo.php

PURPOSE: (1) To allow users to select their origin and destination stations, their
desired start
             and end times, a station that the train should travel through, and
allow users to
             return the information in either a table or a graph.
         (2) To show the tabular information based upon selections made in (1)
         (3) To call the graph visualisation page using selections made in (1)

CREDITS: All code by Matt Griffin.

==================================*/

//Extend PHP's maximum script execution time in order to process the required data
ini_set('max_execution_time', 6000);

//set up MySQL database connection
$con = mysqli_connect("localhost","root","","nrdata");

// Check connection
if (mysqli_connect_errno())
{
       echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

//allocate and clean posted variables to local variables
if((isset($_POST['origin'])) AND (!empty($_POST['origin']))) $originTIPLOC =
$_POST['origin'];
if((isset($_POST['destination'])) AND (!empty($_POST['destination'])))
$destinationTIPLOC = $_POST['destination'];
if((isset($_POST['startTime'])) AND (!empty($_POST['startTime']))) $startingTime =
str_replace(":","",$_POST['startTime']);
if((isset($_POST['endTime'])) AND (!empty($_POST['endTime']))) $endingTime =
str_replace(":","",$_POST['endTime']);
if((isset($_POST['travelVia'])) AND (!empty($_POST['travelVia']))) $travelVia =
$_POST['travelVia'];
if((isset($_POST['daysRun'])) AND (!empty($_POST['daysRun']))) $daysRun =
$_POST['daysRun'];
if((!isset($_POST['travelVia'])) AND (empty($_POST['travelVia']))) $travelVia =
NULL;
if((!isset($_POST['daysRun'])) AND (empty($_POST['daysRun']))) $daysRun = NULL;

//initialise variables for later use
$today = date("Y-m-d");
$dateArray = array();

echo "<HTML><HEAD>";
echo "<link rel='stylesheet' type='text/css' href='stylesheet.css'>";
echo "<TITLE>Multi-Train Information</TITLE></HEAD><BODY>";

//show the selection screen if no selection has been made, or incomplete data was
sent through from the selection screen
if(empty($_POST['origin']) OR empty($_POST['destination']) OR
empty($_POST['startTime']) OR empty($_POST['endTime']))
{
        //display the selection table
```

```
        echo "<CENTER><b>Journey Selection - Please select origin and destination
stations, time span, and days you wish to travel, to view your travel
options</b></center><p>";
        echo "<div class='CSSTableGenerator'>";
       echo "<table border='1'>";
       echo "<tr>";
       echo "<td>Starting Station</td>";
       echo "<td>Ending Station</td>";
       echo "<td>Travel via</td>";
       echo "<td>Time From</td>";
       echo "<td>Time Until</td>";
       echo "<td>Scheduled Days</td>";
       echo "<td colspan='2'></td>";
       echo "</tr>";
       echo "<FORM NAME='requestInfo' METHOD='post'
ACTION='".$_SERVER['PHP_SELF']."'>";

        //pull a list of stations that are stopping points for passengers
       $stationQuery = "SELECT NRTIP.TIPLOCCodeID, NRTIP.TIPLOCDescription FROM
nrtiploc NRTIP STRAIGHT_JOIN nrscheduletrainsignals NRSIG ON (NRTIP.TIPLOCCodeID =
NRSIG.TIPLOCCode) GROUP BY NRTIP.TIPLOCCodeID, NRTIP.TIPLOCDescription ORDER BY
NRTIP.TIPLOCDescription";

       $stationStmt = $con->prepare($stationQuery);
       $stationStmt->execute();
       $stationResult = $stationStmt->get_result();

       echo "<td><select name='origin'>";
        //insert all stations into a drop down selection box
        while ($stationRow = $stationResult->fetch_assoc())
       {
             echo "<option
value='".$stationRow['TIPLOCCodeID']."'>".$stationRow['TIPLOCDescription']."</optio
n>";
       }
       echo "</select></td>";

       $stationQuery = "SELECT NRTIP.TIPLOCCodeID, NRTIP.TIPLOCDescription FROM
nrtiploc NRTIP STRAIGHT_JOIN nrscheduletrainsignals NRSIG ON (NRTIP.TIPLOCCodeID =
NRSIG.TIPLOCCode) GROUP BY NRTIP.TIPLOCCodeID, NRTIP.TIPLOCDescription ORDER BY
NRTIP.TIPLOCDescription";

       $stationStmt = $con->prepare($stationQuery);
       $stationStmt->execute();
       $stationResult = $stationStmt->get_result();

       echo "<td><select name='destination'>";
        //insert all stations into a drop down selection box
       while ($stationRow = $stationResult->fetch_assoc())
       {
             echo "<option
value='".$stationRow['TIPLOCCodeID']."'>".$stationRow['TIPLOCDescription']."</optio
n>";
       }
       echo "</select></td>";

        $stationQuery = "SELECT NRTIP.TIPLOCCodeID, NRTIP.TIPLOCDescription FROM
nrtiploc NRTIP STRAIGHT_JOIN nrscheduletrainsignals NRSIG ON (NRTIP.TIPLOCCodeID =
NRSIG.TIPLOCCode) GROUP BY NRTIP.TIPLOCCodeID, NRTIP.TIPLOCDescription ORDER BY
NRTIP.TIPLOCDescription";

       $stationStmt = $con->prepare($stationQuery);
       $stationStmt->execute();
```

```
	$stationResult = $stationStmt->get_result();

	echo "<td><select name='travelVia'>";
	  //passengers may decide not to select an intermediate station
	  echo "<option value=''></option>";
	  //insert all stations into a drop down selection box
	while ($stationRow = $stationResult->fetch_assoc())
	{
		echo "<option
value='".$stationRow['TIPLOCCodeID']."'>".$stationRow['TIPLOCDescription']."</optio
n>";
	}
	echo "</select></td>";

	  echo "<td><INPUT TYPE='text' NAME='startTime' style='width: 80px;'></td>";
	echo "<td><INPUT TYPE='text' NAME='endTime' style='width: 80px;'></td>";
	echo "<td><SELECT NAME='daysRun'><option
value='1111100'>Weekdays</option><option value='0000010'>Saturdays</option><option
value='0000001'>Sundays</option><option value=''>All Days</option></select></td>";
	echo "<td><INPUT TYPE='Submit' NAME='showTable' VALUE='Show Results'></td>";
	echo "<td><INPUT TYPE='Submit' NAME='showMarey' VALUE='Show Graph'></td>";
	echo "</tr>";
	echo "</FORM></table></div></body></html>";
}
//if users have selected "Show Table" and all data is correctly entered, they will
be shown their selected data in tabular format
else if (isset($_POST['showTable']))
{
	  //allocated posted variables to local variables
	  $originStationTIPLOC = $_POST['origin'];
	$destinationStationTIPLOC = $_POST['destination'];
	$commencementTime = $_POST['startTime'];
	$terminationTime = $_POST['endTime'];
	  if(isset($_POST['travelVia'])) $travelVia = $_POST['travelVia'];

	  //this query will return all services that include the selected starting
and ending stations, as well as intermediate station, for all schedules that are
active
	  $detailQuery = "SELECT NRSCH.scheduleTrainID, NRSCH.signallingID,
NRSCH.daysRun, NRSCH.scheduleStartDate, NRSCH.scheduleEndDate, NRORG.TIPLOCCode AS
originTIPLOC, NRORG.publicDeparture, NRSIG.TIPLOCCode AS destinationTIPLOC,
NRSIG.locationType, NRSIG.publicArrival FROM nrscheduletrains NRSCH INNER JOIN
nrscheduletrainsignals NRSIG ON NRSCH.scheduletrainID = NRSIG.signalTrainID INNER
JOIN (SELECT * FROM nrscheduletrains INNER JOIN nrscheduletrainsignals ON
nrscheduletrains.scheduletrainID = nrscheduletrainsignals.signalTrainID WHERE
nrscheduletrainsignals.locationType <> 'LT' AND nrscheduletrainsignals.TIPLOCCode =
'$originStationTIPLOC' AND nrscheduletrainsignals.signalStartDate <= '$today' AND
nrscheduletrainsignals.signalEndDate >= '$today' ORDER BY
nrscheduletrains.scheduleTrainID ASC, nrscheduletrainsignals.signalPointNo ASC) AS
NRORG ON NRSCH.scheduleTrainID = NRORG.scheduleTrainID AND NRSCH.scheduleStartDate
= NRORG.scheduleStartDate AND NRSCH.scheduleEndDate = NRORG.scheduleEndDate WHERE
NRSIG.locationType <> 'LO' AND NRSIG.TIPLOCCode = '$destinationStationTIPLOC' AND
NRORG.publicDeparture >= '$commencementTime' AND NRORG.publicDeparture <=
'$terminationTime' AND NRSCH.scheduleStartDate <= '$today' AND
NRSCH.scheduleEndDate >= '$today' AND NRSIG.signalStartDate <= '$today' AND
NRSIG.signalEndDate >= '$today' AND NRSIG.signalPointNo > NRORG.signalPointNo AND
NRSIG.publicArrival <> 'pass' AND (NRSCH.trainCategory = 'OO' OR
NRSCH.trainCategory = 'XX') ";
	//if user requests specific days, these will be added to the query string
	  if(isset($daysRun)) $detailQuery .= "AND NRSCH.daysRun = '$daysRun' ";
	$detailQuery .= "ORDER BY NRORG.publicDeparture ASC, NRSCH.scheduleTrainID
ASC, NRSCH.scheduleStartDate ASC";
```

```
	$detailStmt = $con->prepare($detailQuery);
	$detailStmt->execute();
	$detailResult = $detailStmt->get_result();

	  //select the information for the origin and destination stations
	$originQuery = "SELECT * FROM nrtiploc WHERE TIPLOCCodeID =
'".$originStationTIPLOC."'";
	$destinationQuery = "SELECT * FROM nrtiploc WHERE TIPLOCCodeID =
'".$destinationStationTIPLOC."'";

	$originStmt = $con->prepare($originQuery);
	$originStmt->execute();
	$originResult = $originStmt->get_result();

	$destinationStmt = $con->prepare($destinationQuery);
	$destinationStmt->execute();
	$destinationResult = $destinationStmt->get_result();

	  //convert the station names to first letter capitalised
	  while ($originRow = $originResult->fetch_assoc()) {
		while ($destinationRow = $destinationResult->fetch_assoc()) {
			$originStationFullName =
ucwords(strtolower($originRow['TIPLOCDescription']));
			$destinationStationFullName =
ucwords(strtolower($destinationRow['TIPLOCDescription']));
			}
		}

	  //select the date range we wish to extract data from
	  //in this case, the 19th of May is selected as the start date as the summer
schedule was implemented on this date
	$firstDate = new DateTime('2014-05-19');
	$lastDate = new DateTime('2014-08-09');

	while ($firstDate <= $lastDate)
	{
		array_push($dateArray, $firstDate->format('Y-m-d'));
		$firstDate->add(new DateInterval('P1D'));
	}

	echo "<CENTER><b><a
href='NRSingleTrainInfoD3_v6.php?origin=".$originTIPLOC."&destination=".$destinatio
nTIPLOC."&travelVia=".$travelVia."&startTime=".$startingTime."&endTime=".$endingTim
e."&daysRun=".$daysRun."'>Daily train performance - trains departing
$originStationFullName between $commencementTime and $terminationTime and arriving
at $destinationStationFullName (click for graph)</a></center><br>";
	echo "</b></center><p>";
	echo "<div class='CSSTableGenerator'>";
	echo "<table border='1'>";
	echo "<tr>";
	echo "<td>Train ID</td>";
	echo "<td>Signalling ID</td>";
	echo "<td>Days Run</td>";
	echo "<td>Departure<br>($originStationFullName)</td>";
	echo "<td>Arrival<br>($destinationStationFullName)</td>";
	  echo "<td>Average Arrival Time</td>";
	  echo "<td>90% Arrival Time</td>";
	echo "<td>Number of Trains Run</td>";
	echo "<td>Number of Trains Not Run</td>";
	echo "<td>On Time</td>";
	echo "<td>% On Time</td>";
	echo "<td>% within 5m</td>";
	echo "</tr>";
```

```
        //display all information on trains returned from the data set, in tabular
format
       while ($row = $detailResult->fetch_assoc()) {
              $strDays = "";
              echo "<tr>";
              echo "<td><a
href='NRSingleTrainInfoD3_v1.php?serviceID=".$row["scheduleTrainID"]."&dateRun=2014
-05-
19&origin=".$originStationTIPLOC."&destination=".$destinationStationTIPLOC."'>".$ro
w["scheduleTrainID"]."</a></td>";
              echo "<td><a
href='NRSingleTrainInfo.php?signallingID=".$row["signallingID"]."&origin=".$originS
tationTIPLOC."&destination=".$destinationStationTIPLOC."'>".$row["signallingID"]."<
/a></td>";
              $strDaysRun = $row["daysRun"];
              if ($strDaysRun == "1111100") $strDays = "Mon-Fri";
              else
              {
                     if (substr($strDaysRun,0,1) == "1") $strDays .= "Mon/";
                     if (substr($strDaysRun,1,1) == "1") $strDays .= "Tue/";
                     if (substr($strDaysRun,2,1) == "1") $strDays .= "Wed/";
                     if (substr($strDaysRun,3,1) == "1") $strDays .= "Thu/";
                     if (substr($strDaysRun,4,1) == "1") $strDays .= "Fri/";
                     if (substr($strDaysRun,5,1) == "1") $strDays .= "Sat/";
                     if (substr($strDaysRun,6,1) == "1") $strDays .= "Sun/";
                     $strDays = rtrim($strDays, "/");
              }
              echo "<td>".$strDays."</td>";
              echo
"<td>".date('H:i',strtotime(substr($row["publicDeparture"],0,4))."</td>";
              echo
"<td>".date('H:i',strtotime(substr($row["publicArrival"],0,4))."</td>";

                 //return the average arrival time into the destination station, to
the nearest minute
                 $averageQuery = "SELECT
DATE_FORMAT(FROM_UNIXTIME(AVG(UNIX_TIMESTAMP(STR_TO_DATE(RTarrival,'%H%i')))),'%H:%
i') AS 'Actual Arrival' FROM nrhistoricdata NRHIS WHERE NRHIS.trainID =
'".$row["scheduleTrainID"]."' AND NRHIS.trainTIPLOC =
'".$destinationStationTIPLOC."' AND NRHIS.trainStop <> 1 AND NRHIS.RTarrival IS NOT
NULL";
              $averageStmt = $con->prepare($averageQuery);
              $averageStmt->execute();
              $averageResult = $averageStmt->get_result();
                 while ($averageRow = $averageResult->fetch_assoc())
                 {
                     echo "<td>".$averageRow['Actual Arrival']."</td>";
                 }

                 //return the percentage of trains that arrive either on time or
early
                 if ($row["locationType"] == "LO") $percentQuery = "SELECT * FROM
`nrscheduletrains` INNER JOIN `nrscheduletrainsignals` ON
(`nrscheduletrains`.scheduleTrainID = `nrscheduletrainsignals`.signalTrainID) INNER
JOIN `nrhistoricdata` ON (`nrscheduletrains`.scheduleTrainID =
`nrhistoricdata`.trainID AND `nrscheduletrainsignals`.TIPLOCCode =
`nrhistoricdata`.trainTIPLOC AND `nrscheduletrainsignals`.TIPLOCCode =
\"$destinationStationTIPLOC\" AND `nrhistoricdata`.trainTIPLOC =
\"$destinationStationTIPLOC\" AND `nrhistoricdata`.dateRun >=
`nrscheduletrains`.scheduleStartDate AND `nrhistoricdata`.dateRun <=
`nrscheduletrains`.scheduleEndDate) WHERE `nrscheduletrains`.scheduleTrainID =
'".$row["scheduleTrainID"]."' AND `nrhistoricdata`.dateRun >=
```

```
`nrscheduletrainsignals`.signalStartDate AND `nrhistoricdata`.dateRun <=
`nrscheduletrainsignals`.signalEndDate AND `nrhistoricdata`.RTdeparture IS NOT NULL
ORDER BY `nrhistoricdata`.RTarrival ASC";
            elseif ($row["locationType"] == "LT") $percentQuery = "SELECT * FROM
`nrscheduletrains` INNER JOIN `nrscheduletrainsignals` ON
(`nrscheduletrains`.scheduleTrainID = `nrscheduletrainsignals`.signalTrainID) INNER
JOIN `nrhistoricdata` ON (`nrscheduletrains`.scheduleTrainID =
`nrhistoricdata`.trainID AND `nrscheduletrainsignals`.TIPLOCCode =
`nrhistoricdata`.trainTIPLOC AND `nrscheduletrainsignals`.TIPLOCCode =
\"$destinationStationTIPLOC\" AND `nrhistoricdata`.trainTIPLOC =
\"$destinationStationTIPLOC\" AND `nrhistoricdata`.dateRun >=
`nrscheduletrains`.scheduleStartDate AND `nrhistoricdata`.dateRun <=
`nrscheduletrains`.scheduleEndDate) WHERE `nrscheduletrains`.scheduleTrainID =
'".$row["scheduleTrainID"]."' AND `nrhistoricdata`.dateRun >=
`nrscheduletrainsignals`.signalStartDate AND `nrhistoricdata`.dateRun <=
`nrscheduletrainsignals`.signalEndDate AND `nrhistoricdata`.RTarrival IS NOT NULL
ORDER BY `nrhistoricdata`.RTarrival ASC";
            else $percentQuery = "SELECT * FROM `nrscheduletrains` INNER JOIN
`nrscheduletrainsignals` ON (`nrscheduletrains`.scheduleTrainID =
`nrscheduletrainsignals`.signalTrainID) INNER JOIN `nrhistoricdata` ON
(`nrscheduletrains`.scheduleTrainID = `nrhistoricdata`.trainID AND
`nrscheduletrainsignals`.TIPLOCCode = `nrhistoricdata`.trainTIPLOC AND
`nrscheduletrainsignals`.TIPLOCCode = \"$destinationStationTIPLOC\" AND
`nrhistoricdata`.trainTIPLOC = \"$destinationStationTIPLOC\" AND
`nrhistoricdata`.dateRun >= `nrscheduletrains`.scheduleStartDate AND
`nrhistoricdata`.dateRun <= `nrscheduletrains`.scheduleEndDate) WHERE
`nrscheduletrains`.scheduleTrainID = '".$row["scheduleTrainID"]."' AND
`nrhistoricdata`.dateRun >= `nrscheduletrainsignals`.signalStartDate AND
`nrhistoricdata`.dateRun <= `nrscheduletrainsignals`.signalEndDate AND
`nrhistoricdata`.RTarrival IS NOT NULL AND `nrhistoricdata`.RTdeparture IS NOT NULL
ORDER BY `nrhistoricdata`.RTarrival ASC";
            $percentStmt = $con->prepare($percentQuery);
            $percentStmt->execute();
            $percentResult = $percentStmt->get_result();
            $rowCount = $percentResult->num_rows;
               //return the time that 90% of trains have arrived at the
destination.
               //change 'percentageSought' to put whatever figure is desired - 50%
or 95% could also be used for summary data
               $percentageSought = round($rowCount * 0.9);
            $counter = 1;
            if ($rowCount == 0) echo "<td></td>";
            else
            {
                  while ($thisRow = $percentResult->fetch_assoc())
                  {
                        if($counter == $percentageSought)
                        {
                              echo
"<td>".date('H:i',strtotime($thisRow['RTarrival'])).'</td>";
                        }
                        $counter++;
                  }
            }

               //calculate how many trains ran over the selected time period
            $calcQuery = "SELECT scheduleTrainID, STR_TO_DATE(publicArrival,
'%H%i') AS 'public Arrival', COUNT(scheduleTrainID) AS 'Times Run',
SUM(IF(RTarrival IS NULL, 0,IF(STR_TO_DATE(RTarrival,'%H%i') <=
STR_TO_DATE(publicArrival, '%H%i'), 1, 0))) AS 'On Time', SUM(IF(RTarrival IS NULL,
0,IF(ROUND(TIME_TO_SEC(TIMEDIFF(STR_TO_DATE(RTarrival,'%H%i'),STR_TO_DATE(publicArr
ival, '%H%i')))/60 <= 5), 1, 0))) AS 'Times > 5m Late' FROM `nrscheduletrains`
INNER JOIN `nrscheduletrainsignals` ON (`nrscheduletrains`.scheduleTrainID =
```

```
`nrscheduletrainsignals`.signalTrainID AND `nrscheduletrains`.scheduleStartDate =
`nrscheduletrainsignals`.signalStartDate AND `nrscheduletrains`.scheduleEndDate =
`nrscheduletrainsignals`.signalEndDate) INNER JOIN `nrhistoricdata` ON
(`nrscheduletrains`.scheduleTrainID = `nrhistoricdata`.trainID AND
`nrscheduletrainsignals`.TIPLOCCode = `nrhistoricdata`.trainTIPLOC AND
`nrscheduletrainsignals`.TIPLOCCode = '$destinationStationTIPLOC' AND
`nrhistoricdata`.trainTIPLOC = '$destinationStationTIPLOC' AND
`nrhistoricdata`.bookedArrival = `nrscheduletrainsignals`.publicArrival) WHERE
`nrscheduletrains`.scheduleTrainID = '".$row['scheduleTrainID']."' AND dateRun >=
'".$dateArray[0]."' and dateRun <= '".$lastDate->format('Y-m-d')."' AND dateRun >=
`nrscheduletrains`.scheduleStartDate AND dateRun <=
`nrscheduletrains`.scheduleEndDate GROUP BY scheduleTrainID ORDER BY publicArrival
ASC, RTarrival ASC";
                $calcStmt = $con->prepare($calcQuery);
            $calcStmt->execute();
            $calcResult = $calcStmt->get_result();

                //calculate how many trains did not run as scheduled over the
selected time period
            $noServiceQuery = "SELECT trainID, COUNT(trainID) AS 'No Service' FROM
nrhistoricdata WHERE trainID = '".$row['scheduleTrainID']."' AND trainTIPLOC =
'$destinationStationTIPLOC' AND dateRun >= '".$dateArray[0]."' AND dateRun <=
'".$lastDate->format('Y-m-d')."' AND trainCancelReason LIKE '%%' GROUP BY trainID
ORDER BY bookedArrival ASC, RTarrival ASC";
            $noServiceStmt = $con->prepare($noServiceQuery);
            $noServiceStmt->execute();
            $noServiceResult = $noServiceStmt->get_result();

            while ($calcRow = $calcResult->fetch_assoc()) {
                  $rowCountNoService = $noServiceResult->num_rows;
                  if ($rowCountNoService == 0)
                  {
                        $numTimesRun = $calcRow['Times Run'];
                        echo "<td>".$numTimesRun."</td>";
                        echo "<td>0</td>";
                        echo "<td>".$calcRow['On Time']."</td>";
                        echo "<td>".round(($calcRow['On Time'] / $numTimesRun) *
100,1)."%</td>";
                        echo "<td>".round(($calcRow['Times > 5m Late'] /
$numTimesRun) * 100,1)."%</td>";
                  }
                  else
                  {
                        while ($noServiceRow = $noServiceResult->fetch_assoc())
{
                              $numTimesRun = $calcRow['Times Run'] -
$noServiceRow['No Service'];
                              echo "<td>".$numTimesRun."</td>";
                              echo "<td>".$noServiceRow['No Service']."</td>";
                              echo "<td>".$calcRow['On Time']."</td>";
                              echo "<td>".round(($calcRow['On Time'] /
$numTimesRun) * 100,1)."%</td>";
                              echo "<td>".round(($calcRow['Times > 5m Late'] /
$numTimesRun) * 100,1)."%</td>";
                        }
                  }
            }

            echo "</tr>";
      }
      echo "</table></div></body></html>";
}
```

```
//if users have selected "Show Graph", then convert the posted variables to local
variables and call the Marey graph page
//we need to to convert the variables as they are not automatically passed through
to the included Marey graph page
else if(isset($_POST['showMarey']))
{
    $_GET['origin'] = $originTIPLOC;
    $_GET['destination'] = $destinationTIPLOC;
    $_GET['startTime'] = $startingTime;
    $_GET['endTime'] = $endingTime;
    $_GET['travelVia'] = $travelVia;
    include 'NRSingleTrainInfoD3_v6.php';
}
?>
```

Page: NRSchedule_parserCLI.php

```
<?php

/*==================================
FILE: NRSchedule_parserCLI.php

PURPOSE: (1) Process the National Rail schedule, identify trains we wish to include,
and extract the relevant information
         (2) Calculate links between stations for the purposes of finding a path
from unrelated stops
         (3) Process additional fixed links from a seperate file

CREDITS: All work by Matt Griffin

==================================*/

//Extend PHP's maximum script execution time in order to process the required data
ini_set('max_execution_time', 6000);
ini_set('memory_limit', '512M');

//set up MySQL database connection
$con = mysqli_connect("localhost","root","","nrdata");

// Check connection
if (mysqli_connect_errno())
{
       echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

//create a blank array to accept a pair of links between stops
$pairArray = array(
                                    "linkType" => "BUS",
                                    "CRSOrigin" => NULL,
                                    "CRSDestination" => NULL,
                                    "originDeptTime" => NULL,
                                    "destinationDeptTime" => NULL,
                                    "linkMinutes" => NULL,
                                    );

//initialise train and signal queries. these will be added to as the script
continues processing train data
$trainQuery = "INSERT INTO NRScheduleTrains (scheduleTrainID, scheduleType,
trainOperator, trainCategory, daysRun, signallingID, scheduleStartDate,
scheduleEndDate)";
$trainQuery .= " VALUES ";
$signalQuery = "INSERT INTO NRScheduleTrainSignals (signalTrainID, signalStartDate,
signalEndDate, locationType, signalPointNo, TIPLOCCode, internalArrival,
internalDeparture, publicArrival, publicDeparture)";
$signalQuery .= " VALUES ";
$tiplocQuery = "INSERT INTO NRtiploc (TIPLOCCodeID, crsCode, TIPLOCDescription)";
$tiplocQuery .= " VALUES ";

//select the schedule source file. this must be in the same directory as the system
source files
//in this case, we are loading just one schedule at a time
$src = __DIR__ . '/CIF_HU_TOC_FULL_DAILY-toc-full';

//initialise countStations and string variables for future use
$countStations = 0;
$string = "";

//open the source file for processing
```

```
$handle = @fopen($src, "r");
//if the file exists, execute the following code
if ($handle) {
       //process the file while there is data
        while (($buffer = stream_get_line($handle, 65535, "\n")) !== false) {
        $countStations = 0;
        //convert the string that has been read in to a JSON string
        $json_result = json_decode($buffer, true);

        //if the string is a station (TIPLOC) identifier, parse the string and add
this station to the NRTIPLOC table
        if(!empty($json_result['TiplocV1']))
        {
            //some stations include special characters. these must be handled
appropriately
            $tiplocQuery .=
"('".addslashes($json_result['TiplocV1']['tiploc_code'])."', ";
            $tiplocQuery .= "'".addslashes($json_result['TiplocV1']['crs_code'])."',
";
            $tiplocQuery .=
"'".addslashes($json_result['TiplocV1']['tps_description'])."'),";
        }
        else if(empty($json_result['TiplocV1'])) echo "";

        //this next set of checks determines which schedules we want to include in
our data set
        //first, check if the schedule information returns any valid data
       if(empty($json_result['JsonScheduleV1'])) echo "";
        //then, ignore obsolete ("C") or overwritten ("O") schedules as newer
schedules will be captured later
       else if(($json_result['JsonScheduleV1']['CIF_stp_indicator'] == "C") OR
($json_result['JsonScheduleV1']['CIF_stp_indicator'] == "O")) echo "";
        //then, ignore any bus replacement schedules as these cannot be accurately
captured
       else
if(($json_result['JsonScheduleV1']['schedule_segment']['CIF_train_category'] ==
"BR")) echo "";
        //finally, narrow down the train operators; in this case, Southern,
SouthWest, Southeastern, First Capital Connect, and Eurostar
       else if(($json_result['JsonScheduleV1']['atoc_code'] == "ES") OR
($json_result['JsonScheduleV1']['atoc_code'] == "EM") OR
($json_result['JsonScheduleV1']['atoc_code'] == "FC") OR
($json_result['JsonScheduleV1']['atoc_code'] == "SN") OR
($json_result['JsonScheduleV1']['atoc_code'] == "GX") OR
($json_result['JsonScheduleV1']['atoc_code'] == "SE") OR
($json_result['JsonScheduleV1']['atoc_code'] == "SW"))
       {
                  //initialise the variables for the station link array
              $pairArray['crsOrigin'] = NULL;
              $pairArray['crsDestination'] = NULL;
              $pairArray['originDeptTime'] = NULL;
              $pairArray['destinationDeptTime'] = NULL;
              $pairArray['linkMinutes'] = NULL;

                  //it's possible that a scheduled train has no stops - ignore these

       if(empty($json_result['JsonScheduleV1']['schedule_segment']['schedule_locati
on'])) echo "";
              else
              {
                        //loop through each schedule location in the string
```

```
      foreach($json_result['JsonScheduleV1']['schedule_segment']['schedule_locatio
n'] as $value)
                    {
                                  //by keeping track of each station, we can more
easily identify the route
                           $countStations++;
                           //This adds the previous station together to the current
station to pair them as a valid train route,
                           //ignoring any stops that are the route origin (as they
have no prior stops or "nodes")
                           if (($value['location_type'] == "LT") OR
(($value['location_type'] == "LI") && (!is_null($value['public_departure']))))
                           {
                                        //this works out the next station along
from the station, calculates how many minutes it would take to transit
                                        //between these two stations, then add them
to the array
                                  $pairArray['crsDestination'] =
$value['tiploc_code'];
                                  $pairArray['destinationDeptTime'] =
$value['public_arrival'];
                                  $timeEnd =
substr($pairArray['destinationDeptTime'],0,2).":".substr($pairArray['destinationDep
tTime'],-2);
                                  $timeStart =
substr($pairArray['originDeptTime'],0,2).":".substr($pairArray['originDeptTime'],-
2);
                                  $timeEnd = DateTime::createFromFormat('H:i',
$timeEnd);
                                  $timeStart = DateTime::createFromFormat('H:i',
$timeStart);
                                  if(isset($pairArray['originDeptTime']) &&
isset($pairArray['destinationDeptTime']))
                                  {
                                                 //allow for times that overlap into
the next hour - no sense having 63 minutes between stops if it's only 3!
                                                 if (($timeStart->format('H') >= 1)
&& ($timeEnd->format('H') <= 1))
                                        {
                                               $timeLeft =
date_diff($timeStart,$timeEnd);
                                               $pairArray['linkMinutes'] = 1400 -
((($timeLeft->format('%h') + 1) * 60) - $timeLeft->format('%i'));
                                        }
                                        else
                                        {
                                               $timeLeft =
date_diff($timeStart,$timeEnd);
                                               $pairArray['linkMinutes'] =
(($timeLeft->format('%h') * 60) + $timeLeft->format('%i'));
                                        }
                                        $routeQuery = "INSERT INTO NRFixedLinks
(linkType, CRSOrigin, CRSDestination, linkMinutes)";
                                        $routeQuery .= " VALUES ";
                                        $routeQuery .= "('TRAIN', ";
                                        $routeQuery .=
"'".$pairArray['crsOrigin']."', ";
                                        $routeQuery .=
"'".$pairArray['crsDestination']."', ";
                                        $routeQuery .=
"".$pairArray['linkMinutes'].")";
```

```
                                        if(substr($routeQuery, -6) == "VALUES")
echo "";
                                        else
                                        {
                                               $routeQuery = rtrim($routeQuery, ",
");
                                               $stmtRoute = $con-
>prepare($routeQuery);
                                               $stmtRoute->execute();
                                               $result = $stmtRoute->get_result();
                                        }

                                 }

                                 $pairArray['crsOrigin'] =
$pairArray['crsDestination'];
                                 $pairArray['crsDestination'] = NULL;
                                 if ($value['location_type'] == "LI")
                                 {
                                        $pairArray['originDeptTime'] =
$value['public_departure'];
                                 }
                                 $pairArray['destinationDeptTime'] = NULL;
                                 $pairArray['linkMinutes'] = NULL;


                           }

                                 //capture information regarding the train service
itself to include in the signal query
                                 //this allows us to match up trains to their stops
later in the system
                           $signalQuery .=
"('".$json_result['JsonScheduleV1']['CIF_train_uid']."', ";
                           $signalQuery .=
"'".$json_result['JsonScheduleV1']['schedule_start_date']."', ";
                           $signalQuery .=
"'".$json_result['JsonScheduleV1']['schedule_end_date']."', ";
                           $signalQuery .= "'".$value['location_type']."', ";
                           $signalQuery .= $countStations.", ";
                           $signalQuery .= "'".$value['tiploc_code']."', ";

                                 if ($value['location_type'] == "LO")
                           {
                                  $pairArray['crsOrigin'] = $value['tiploc_code'];
                                  $pairArray['originDeptTime'] =
$value['public_departure'];
                           }
                                 //some journeys will stop at every station. some
journeys do not stop at all stations.
                                 //we need to prepare the query to accept either a
time or 'pass', as in, 'train passed without stopping'
                                 if((empty($value['arrival'])) &&
(empty($value['pass']))) $signalQuery .= "NULL, ";
                                  elseif((empty($value['arrival'])) &&
(!empty($value['pass']))) $signalQuery .= "'pass', ";
                                  else $signalQuery .= "'".$value['arrival']."', ";
                           if((empty($value['departure'])) &&
(empty($value['pass']))) $signalQuery .= "NULL, ";
                                  elseif((empty($value['departure'])) &&
(!empty($value['pass']))) $signalQuery .= "'".$value['pass']."', ";
                                 else $signalQuery .= "'".$value['departure']."',
";
```

```
                            if(empty($value['public_arrival'])) $signalQuery .=
"NULL, ";
                            else $signalQuery .= "'".$value['public_arrival']."', ";
                            if(empty($value['public_departure'])) $signalQuery .=
"NULL),";
                            else $signalQuery .=
"'".$value['public_departure']."'),";
                        }
                            //capture information regarding the train itself, as well
as the schedule that is being analysed
                            //some trains will have more than one schedule, due to
operational reasons
                            //some trains may only run on one day (Bank Holidays,
special services, etc.)
                        $trainQuery .=
"('".$json_result['JsonScheduleV1']['CIF_train_uid']."', ";
                        $trainQuery .=
"'".$json_result['JsonScheduleV1']['CIF_stp_indicator']."', ";
                        $trainQuery .=
"'".$json_result['JsonScheduleV1']['atoc_code']."', ";
                        $trainQuery .=
"'".$json_result['JsonScheduleV1']['schedule_segment']['CIF_train_category']."', ";
                        $trainQuery .=
"'".$json_result['JsonScheduleV1']['schedule_days_runs']."', ";
                        $trainQuery .=
"'".$json_result['JsonScheduleV1']['schedule_segment']['signalling_id']."',";
                        $trainQuery .=
"'".$json_result['JsonScheduleV1']['schedule_start_date']."', ";
                        $trainQuery .=
"'".$json_result['JsonScheduleV1']['schedule_end_date']."'),";

                }
        }
        else echo "No file found";

        }
//error handling for the schedule file if there is an issue with the EOF symbol
if (!feof($handle)) {
        echo "Error: unexpected fgets() fail\n";
}

//prepare the query using PHP's inbuilt functionality, then execute and check for
result
$tiplocQuery = rtrim($tiplocQuery, ",");
$stmtTIPLOC = $con->prepare($tiplocQuery);
$stmtTIPLOC->execute();
$result = $stmtTIPLOC->get_result();

//prepare the train and signal queries by removing the trailing comma
$trainQuery = rtrim($trainQuery, ",");
$signalQuery = rtrim($signalQuery, ",");

//prepare the queries for execution, then execute them and return the result
$stmtTrain = $con->prepare($trainQuery);
$stmtTrain->execute();
$result = $stmtTrain->get_result();

$stmtSignal = $con->prepare($signalQuery);
$stmtSignal->execute();
$result = $stmtSignal->get_result();

//close the schedule file
fclose($handle);
```

```
}

//this section deals with the fixed links file
//first, open the file; again, this must be in the same folder at this source file
$src = __DIR__ . '/Master Station Names.txt';

//initialise station variables
$stationTIPLOC = "";
$stationCRS = "";
$stationPassengerStop = "";

//open the file
$handle = @fopen($src, "r");

if ($handle) {
       while (($buffer = stream_get_line($handle, 65535, "\n")) !== false) {
            //ignore the opening file line
            if(substr($buffer,0,1) == "A")
            {
                //ignore the FILE SPEC line as it contains no useful data
                if(substr($buffer,30,10) == "FILE-SPEC=")
                {
                    echo "";
                }
                else
                {
                    //process each line according to the National Rail handbook
                    $stationTIPLOC = trim(substr($buffer,36,7));
                    $stationCRS = substr($buffer,43,3);
                    $tiplocQuery = "UPDATE NRtiploc SET crsCode = '$stationCRS',
passengerStop = 'Y' WHERE TIPLOCCodeID = '$stationTIPLOC'";
                    $stmtTIPLOCUpdate = $con->prepare($tiplocQuery);
                    $stmtTIPLOCUpdate->execute();
                    $result = $stmtTIPLOCUpdate->get_result();
                }
            }
        }
}
//close file
fclose($handle);

//this section deals with the file containing latitude and longitude
//first, open the file; again, this must be in the same folder at this source file
$src = __DIR__ . '/ukrail_locations.csv';

$handle = @fopen($src, "r");
while (!feof($handle) ) {
    while (($buffer = stream_get_line($handle, 1024, "\n")) !== false) {
        $stationInfo = str_getcsv($buffer);
        if(!empty($stationInfo[3]) && $stationInfo[8] != "null" &&
$stationInfo[9] != "null" )
        {
            $latlonQuery = "UPDATE NRtiploc SET latitude = ".$stationInfo[9].",
longitude = ".$stationInfo[8]." WHERE TIPLOCCodeID = '$stationInfo[3]'";
            echo $latlonQuery."<br>";
            $latlonUpdate = $con->prepare($latlonQuery);
            $latlonUpdate->execute();
            $latlonResult = $latlonUpdate->get_result();
        }
    }
}
fclose($handle);
?>
```

Page: NRSingleServiceInfo.php

```
<?php

/*==================================
FILE: NRSingleServiceInfo.php

PURPOSE: To return information on a single train journey in tabular format. Allows
a link to the graph visualisation.

CREDITS: All work by Matt Griffin

==================================*/

//Extend PHP's maximum script execution time in order to process the required data
ini_set('max_execution_time', 6000);

//connect to the database
$con = mysqli_connect("localhost","root","","nrdata");

// Check connection
if (mysqli_connect_errno())
{
	echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

//allocate posted variables to local variables
$serviceID = $_GET['serviceID'];
$dateRun = $_GET['dateRun'];
$originTIPLOC = $_GET['origin'];
$destinationTIPLOC = $_GET['destination'];

//return the starting and ending points, plus the departure time at the starting
station and the arrival time at the ending station
//used for the page header
$headerTerminationQuery =  "SELECT NRTIP.TIPLOCDescription AS 'Origin',
NRSIG.publicDeparture, NRTIP2.TIPLOCDescription AS 'Destination',
NRSIG2.publicArrival FROM nrscheduletrains NRSCH INNER JOIN nrscheduletrainsignals
NRSIG ON (NRSCH.scheduleTrainID = NRSIG.signalTrainID AND NRSIG.locationType <>
\"LT\" AND NRSIG.TIPLOCCode = '$originTIPLOC') INNER JOIN nrscheduletrainsignals
NRSIG2 ON (NRSCH.scheduleTrainID = NRSIG2.signalTrainID AND NRSIG2.locationType <>
\"LO\" AND NRSIG2.TIPLOCCode = '$destinationTIPLOC') INNER JOIN nrtiploc NRTIP ON
(NRSIG.TIPLOCCode = NRTIP.TIPLOCCodeID) INNER JOIN nrtiploc NRTIP2 ON
(NRSIG2.TIPLOCCode = NRTIP2.TIPLOCCodeID) WHERE NRSCH.scheduleTrainID =
'$serviceID' LIMIT 1";
$headerTerminationStmt = $con->prepare($headerTerminationQuery);
$headerTerminationStmt->execute();
$headerTerminationResult = $headerTerminationStmt->get_result();

echo "<HTML><HEAD>";
echo "<link rel='stylesheet' type='text/css' href='stylesheet.css'>";
echo "<TITLE>Service Information</TITLE></HEAD><BODY>";

//format the station names for cleaner presentation
while ($terminationRow = $headerTerminationResult->fetch_assoc()) {
	$originTime = $terminationRow['publicDeparture'];
	$originStation = ucwords(strtolower($terminationRow['Origin']));
	$destinationTime = $terminationRow['publicArrival'];
	$destinationStation = ucwords(strtolower($terminationRow['Destination']));
}

//page header, table header
```

```
echo "<CENTER><b>Information for the train service from ".$originStation."
(departure ".$originTime.") to ".$destinationStation." (arrival
".$destinationTime.") - run on ".$dateRun;
echo "</b></center><p>";
echo "<center><a
href='NRSingleTrainInfoD3_v4.php?serviceID=".$serviceID."&dateRun=".$dateRun."&orig
in=".$originTIPLOC."&destination=".$destinationTIPLOC."'>Show Marey
chart</a></center><p>";
echo "<div class='CSSTableGenerator'>";
echo "<table border='1'>";
echo "<tr>";
echo "<td>Train Operator</td>";
echo "<td>Train ID</td>";
echo "<td>Signalling ID</td>";
echo "<td>Train Stop</td>";
echo "<td>Station Name</td>";
echo "<td>Scheduled Arrival</td>";
echo "<td>Actual Arrival</td>";
echo "<td>Scheduled Departure</td>";
echo "<td>Actual Departure</td>";
echo "<td>Minutes Late</td>";
echo "<td>Reason for Cancellation</td>";
echo "</tr>";

//return information on all stops for the single journey
//used for the information table
$detailQuery = "SELECT * FROM `nrscheduletrains` INNER JOIN `nrhistoricdata` ON
(`nrscheduletrains`.scheduleTrainID = `nrhistoricdata`.trainID AND
`nrhistoricdata`.dateRun >= `nrscheduletrains`.scheduleStartDate AND
`nrhistoricdata`.dateRun <= `nrscheduletrains`.scheduleEndDate) WHERE
`nrscheduletrains`.scheduleTrainID = \"$serviceID\" AND
`nrhistoricdata`.dateRun='$dateRun' ORDER BY `nrhistoricdata`.dateRun ASC";
$detailStmt = $con->prepare($detailQuery);
$detailStmt->execute();
$detailResult = $detailStmt->get_result();

//loop through each row of the result set and output in tabular format
while ($row = $detailResult->fetch_assoc()) {

       echo "<tr>";
       echo "<td>".$row["trainOperator"]."</td>";
       echo "<td>".$row["scheduleTrainID"]."</td>";
       echo "<td>".$row["signallingID"]."</td>";
       echo "<td>".$row["trainStop"]."</td>";
       echo "<td>".$row["trainTIPLOCDescription"]."</td>";
        //convert any times marked as "H" (30 seconds) to straightforward
hours:minutes for display purposes
        //check for blank arrival and departure times - there may be more than one
reason why this is so (cancellation, no timing data, no timing point, etc.)
        if (empty($row["bookedArrival"])) echo "<td></td>";
              elseif (substr($row["bookedArrival"],4,1) == "H") echo
"<td>".substr($row["bookedArrival"],0,2).":".substr($row["bookedArrival"],2,2)."<
/td>";
              else echo
"<td>".substr($row["bookedArrival"],0,2).":".substr($row["bookedArrival"],-
2)."</td>";
       if (empty($row["RTarrival"])) echo "<td></td>";
              else echo
"<td>".substr($row["RTarrival"],0,2).":".substr($row["RTarrival"],-2)."</td>";
       if (empty($row["bookedDeparture"])) echo "<td></td>";
              elseif (substr($row["bookedDeparture"],4,1) == "H") echo
"<td>".substr($row["bookedDeparture"],0,2).":".substr($row["bookedDeparture"],2,2).
"</td>";
```

```
            else echo
"<td>".substr($row["bookedDeparture"],0,2).":".substr($row["bookedDeparture"],-
2)."</td>";
       if (empty($row["RTdeparture"])) echo "<td></td>";
            else echo
"<td>".substr($row["RTdeparture"],0,2).":".substr($row["RTdeparture"],-2)."</td>";
       if (empty($row["RTlateness"])) echo "<td></td>";
            else echo "<td>".$row["RTlateness"]."</td>";
        //if there is a reason why the train was cancelled, return it here
        //not all trains are cancelled at the beginning of the journey. it is
possible that they are cancelled en route
        //also note that trains that are "cancelled" may be reinstated, but the
reason persists through the train journey
        echo "<td>".$row["trainCancelReason"]."</td>";
       echo "</tr>";
}
echo "</table></div>";

?>
```

Page: NRSingleTrainInfo.php

```
<?php

/*==================================
FILE: NRSingleTrainInfo.php

PURPOSE: To return all journeys from a single signalling ID, regardless of day run.
This allows commuters and railway staff to
         look at all historic journeys for the scheduled train.

EXPLANATION: The signalling ID is equivalent to the headcode of the train. This is
from the days when trains carried the code
              on the front of the train (some services still do).

CREDITS: All work by Matt Griffin

==================================*/

//Extend PHP's maximum script execution time in order to process the required data
ini_set('max_execution_time', 6000);

//connect to MySQL database
$con = mysqli_connect("localhost","root","","nrdata");

// Check connection
if (mysqli_connect_errno())
{
       echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

//allocated passed variables to local variables
$scheduleTrainID = $_GET['scheduleTrainID'];
$originTIPLOC = $_GET['origin'];
$destinationTIPLOC = $_GET['destination'];

//select the journey details from the scheduled train ID
$headerTerminationQuery =  "SELECT NRTIP.TIPLOCDescription AS 'Origin',
NRSIG.publicDeparture, NRTIP2.TIPLOCDescription AS 'Destination',
NRSIG2.publicArrival FROM nrscheduletrains NRSCH INNER JOIN nrscheduletrainsignals
NRSIG ON (NRSCH.scheduleTrainID = NRSIG.signalTrainID AND NRSIG.locationType <>
\"LT\" AND NRSIG.TIPLOCCode = '$originTIPLOC') INNER JOIN nrscheduletrainsignals
NRSIG2 ON (NRSCH.scheduleTrainID = NRSIG2.signalTrainID AND NRSIG2.locationType <>
\"LO\" AND NRSIG2.TIPLOCCode = '$destinationTIPLOC') INNER JOIN nrtiploc NRTIP ON
(NRSIG.TIPLOCCode = NRTIP.TIPLOCCodeID) INNER JOIN nrtiploc NRTIP2 ON
(NRSIG2.TIPLOCCode = NRTIP2.TIPLOCCodeID) WHERE NRSCH.scheduleTrainID =
'$scheduleTrainID' LIMIT 1";
//select the arrival details at the destination for these journeys
$detailQuery = "SELECT * FROM `nrscheduletrains` INNER JOIN
`nrscheduletrainsignals` ON (`nrscheduletrains`.scheduleTrainID =
`nrscheduletrainsignals`.signalTrainID) INNER JOIN `nrhistoricdata` ON
(`nrscheduletrains`.scheduleTrainID = `nrhistoricdata`.trainID AND
`nrscheduletrainsignals`.TIPLOCCode = `nrhistoricdata`.trainTIPLOC AND
`nrscheduletrainsignals`.TIPLOCCode = \"$destinationTIPLOC\" AND
`nrhistoricdata`.trainTIPLOC = \"$destinationTIPLOC\" AND `nrhistoricdata`.dateRun
>= `nrscheduletrains`.scheduleStartDate AND `nrhistoricdata`.dateRun <=
`nrscheduletrains`.scheduleEndDate) WHERE `nrscheduletrains`.scheduleTrainID =
\"$scheduleTrainID\" ORDER BY `nrhistoricdata`.dateRun ASC,
`nrscheduletrainsignals`.signalPointNo ASC";

//prepare, then execute these queries
$headerTerminationStmt = $con->prepare($headerTerminationQuery);
$headerTerminationStmt->execute();
```

```
$headerTerminationResult = $headerTerminationStmt->get_result();

$detailStmt = $con->prepare($detailQuery);
$detailStmt->execute();
$detailResult = $detailStmt->get_result();

//build the HTML header
echo "<HTML><HEAD>";
echo "<script src='http://d3js.org/d3.v3.min.js' charset='utf-8'></script>";
echo "<link rel='stylesheet' type='text/css' href='stylesheet.css'>";
echo "<TITLE>Single Train Information</TITLE></HEAD><BODY>";

//convert the station names into something a little more presentable (i.e., first
letter capitalised)
while ($terminationRow = $headerTerminationResult->fetch_assoc()) {
       $originTime = $terminationRow['publicDeparture'];
       $originStation = ucwords(strtolower($terminationRow['Origin']));
       $destinationTime = $terminationRow['publicArrival'];
       $destinationStation = ucwords(strtolower($terminationRow['Destination']));
}
//build the page title and the table header
echo "<CENTER><b>Daily information for the train from ".$originStation." (departure
".$originTime.") to ".$destinationStation." (arrival ".$destinationTime.") - all
movement at ".$destinationStation;
echo "</b></center><p>";
echo "<div class='CSSTableGenerator'>";
echo "<table border='1'>";
echo "<tr>";
echo "<td>Train ID</td>";
echo "<td>Signalling ID</td>";
echo "<td>Train Operator</td>";
echo "<td>Days Run</td>";
echo "<td>Location Type</td>";
echo "<td>Date Run</td>";
echo "<td>Stop Number</td>";
echo "<td>TIPLOC Code</td>";
echo "<td>WTT Arrival</td>";
echo "<td>WTT Departure</td>";
echo "<td>Public Arrival</td>";
echo "<td>Public Departure</td>";
echo "<td>Actual Arrival</td>";
echo "<td>Actual Departure</td>";
echo "<td>Reason for Cancellation</td>";
echo "</tr>";

//loop through all journey results
while ($row = $detailResult->fetch_assoc()) {
        //display key information on the train service
       echo "<tr>";
       echo "<td>".$row["scheduleTrainID"]."</td>";
       echo "<td>".$row["signallingID"]."</td>";
       echo "<td>".$row["trainOperator"]."</td>";
       $strDays = "";
        //indicate to commuters/staff what days this train service has run
        if (($row["daysRun"][0]=="1") && ($row["daysRun"][1]=="1") &&
($row["daysRun"][2]=="1") &&($row["daysRun"][3]=="1") && ($row["daysRun"][4]=="1")
&& ($row["daysRun"][5]=="0") && ($row["daysRun"][6]=="0")) $strDays = "Mon-Fri";
       else
       {
              if ($row["daysRun"][0]=="1") $strDays .= "Mon/";
              if ($row["daysRun"][1]=="1") $strDays .= "Tue/";
              if ($row["daysRun"][2]=="1") $strDays .= "Wed/";
              if ($row["daysRun"][3]=="1") $strDays .= "Thu/";
```

```
            if ($row["daysRun"][4]=="1") $strDays .= "Fri/";
            if ($row["daysRun"][5]=="1") $strDays .= "Sat/";
            if ($row["daysRun"][6]=="1") $strDays .= "Sun/";
            $strDays = rtrim($strDays, "/");
      }
      echo "<td>".$strDays."</td>";
      echo "<td>".$row["locationType"]."</td>";
      $obj_date = DateTime::createFromFormat('Y-m-d', $row["dateRun"]);
      echo "<td>".$obj_date->format('d/m/Y')."</td>";
      echo "<td>".$row["signalPointNo"]."</td>";
      echo "<td>".$row["trainTIPLOCDescription"]."</td>";
       //convert times to insert a colon between hours and minutes for better
presentation
       //also convert times that are suffixed as "H" (indicating arrival on 30
seconds past the minute) to round to nearest minute
       if (empty($row["internalArrival"])) echo "<td></td>";
            elseif (substr($row["internalArrival"],4,1) == "H") echo
"<td>".substr($row["internalArrival"],0,2).":".substr($row["internalArrival"],2,2).
"�</td>";
            else echo
"<td>".substr($row["internalArrival"],0,2).":".substr($row["internalArrival"],-
2)."</td>";
      if (empty($row["internalDeparture"])) echo "<td></td>";
            elseif (substr($row["internalDeparture"],4,1) == "H") echo
"<td>".substr($row["internalDeparture"],0,2).":".substr($row["internalDeparture"],2
,2)."�</td>";
            else echo
"<td>".substr($row["internalDeparture"],0,2).":".substr($row["internalDeparture"],-
2)."</td>";
      if (empty($row["publicArrival"])) echo "<td></td>";
            else echo
"<td>".substr($row["publicArrival"],0,2).":".substr($row["publicArrival"],-
2)."</td>";
      if (empty($row["publicDeparture"])) echo "<td></td>";
            else echo
"<td>".substr($row["publicDeparture"],0,2).":".substr($row["publicDeparture"],-
2)."</td>";
      if (empty($row["RTarrival"])) echo "<td></td>";
            else echo
"<td>".substr($row["RTarrival"],0,2).":".substr($row["RTarrival"],-2)."</td>";
      if (empty($row["RTdeparture"])) echo "<td></td>";
            else echo
"<td>".substr($row["RTdeparture"],0,2).":".substr($row["RTdeparture"],-2)."</td>";
       //display reason for train cancellation (if train was, indeed, cancelled)
       echo "<td>".ucfirst($row["trainCancelReason"])."</td>";
      echo "</tr>";
}
echo "</table></div>";

?>
```

Page: NRSingleTrainInfoD3_v6.php

```
<?php

/*==================================
FILE: NRSingleTrainInfoD3.php

PURPOSE: To display all journeys for the train schedules returned from the user
selections.
         Using a combination of Javascript and D3, each journey is plotted on the
screen, with the scheduled train and average
         train plotted as black and blue lines respectively.

CREDITS: Mike Bostock for his Marey graph template - this work adapts from the
original template at http://bl.ocks.org/mbostock/5544008
         Reindexing of JS array found here ->
http://stackoverflow.com/questions/4759745/javascript-reindexing-an-array
         Code for converting time to 12 hour AM/PM found here ->
http://stackoverflow.com/questions/8888491/how-do-you-display-javascript-datetime-
in-12-hour-am-pm-format
         Code for calculating distance using latitude/longitude here ->
http://www.movable-type.co.uk/scripts/latlong.html
         Code to round the time to nearest 15 minutes for axes here ->
http://stackoverflow.com/questions/4968250/how-to-round-time-to-the-nearest-
quarter-hour-in-javascript

==================================*/

//Extend PHP's maximum script execution time in order to process the required data
ini_set('max_execution_time', 6000);

$con = mysqli_connect("localhost","root","","nrdata");

// Check connection
if (mysqli_connect_errno())
{
      echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

//allocated posted variables to local variables
$originTIPLOC = $_GET['origin'];
$destinationTIPLOC = $_GET['destination'];
$startingTime = str_replace(":","",$_GET['startTime']);
$endingTime = str_replace(":","",$_GET['endTime']);
if((isset($_GET['daysRun'])) AND (!empty($_GET['daysRun']))) $daysRun =
$_GET['daysRun'];
if((isset($_GET['travelVia'])) AND (!empty($_GET['travelVia']))) $viaTIPLOC =
$_GET['travelVia'];

$today = date("Y-m-d");

//retrieve details on all journeys matching the criteria selected by user
$detailQuery = "SELECT NRSCH.scheduleTrainID, NRSCH.signallingID, NRSCH.daysRun,
NRSCH.scheduleStartDate, NRSCH.scheduleEndDate, NRORG.signalPointNo AS startNumber,
NRORG.TIPLOCCode AS originTIPLOC, NRORG.publicDeparture, NRSIG.signalPointNo As
endNumber, NRSIG.TIPLOCCode AS destinationTIPLOC, NRSIG.locationType,
NRSIG.publicArrival FROM nrscheduletrains NRSCH INNER JOIN nrscheduletrainsignals
NRSIG ON NRSCH.scheduletrainID = NRSIG.signalTrainID INNER JOIN (SELECT * FROM
nrscheduletrains INNER JOIN nrscheduletrainsignals ON
nrscheduletrains.scheduletrainID = nrscheduletrainsignals.signalTrainID WHERE
nrscheduletrainsignals.locationType <> 'LT' AND nrscheduletrainsignals.TIPLOCCode =
'$originTIPLOC' AND nrscheduletrainsignals.signalStartDate <= '$today' AND
nrscheduletrainsignals.signalEndDate >= '$today' ORDER BY
```

```
nrscheduletrains.scheduleTrainID ASC, nrscheduletrainsignals.signalPointNo ASC) AS
NRORG ON NRSCH.scheduleTrainID = NRORG.scheduleTrainID AND NRSCH.scheduleStartDate
= NRORG.scheduleStartDate AND NRSCH.scheduleEndDate = NRORG.scheduleEndDate WHERE
NRSIG.locationType <> 'LO' AND NRSIG.TIPLOCCode = '$destinationTIPLOC' AND
NRORG.publicDeparture >= '$startingTime' AND NRORG.publicDeparture <= '$endingTime'
AND NRSCH.scheduleStartDate <= '$today' AND NRSCH.scheduleEndDate >= '$today' AND
NRSIG.signalStartDate <= '$today' AND NRSIG.signalEndDate >= '$today' AND
NRSIG.signalPointNo > NRORG.signalPointNo AND NRSIG.publicArrival <> 'pass' AND
(NRSCH.trainCategory = 'OO' OR NRSCH.trainCategory = 'XX') ";
if(isset($daysRun)) $detailQuery .= "AND NRSCH.daysRun = '$daysRun' ";
$detailQuery .= "ORDER BY NRORG.publicDeparture ASC, NRSCH.scheduleTrainID ASC,
NRSCH.scheduleStartDate ASC";
$detailStmt = $con->prepare($detailQuery);
$detailStmt->execute();
$detailResult = $detailStmt->get_result();

echo "<HTML><HEAD>";
//D3 library is added using the online library. This means an active Internet
connection is required to use this file.
echo "<script src='http://d3js.org/d3.v3.min.js' charset='utf-8'></script>";
echo "<link rel='stylesheet' type='text/css' href='stylesheet.css'>";
echo "<TITLE>Multiple Train Information</TITLE></HEAD>";
//declare the Javascript formatting required to display journey information
?>
<style>

svg {
  font: 10px sans-serif;
}

.line {
  fill: none;
  stroke: url(#gradient);
  stroke-width: 4.5px;
}

.axis path {
  display: none;
}

.axis line {
  stroke: #000;
  shape-rendering: crispEdges;
}

.station line {
  stroke: #ddd;
  stroke-dasharray: 1,1;
  shape-rendering: geometricPrecision;
}

.station text {
  text-anchor: end;
}

.train .scheduled line {
  fill: none;
  stroke-width: 3px;
}

.train .actual line {
  fill: none;
  stroke-width: 3px;
```

```
}

.train .scheduled circle {
  stroke: #fff;
}

.train .actual circle {
  stroke: #fff;
}

.train .scheduled path { stroke: rgb(0,0,0); }

.train .actual path { stroke: rgb(138,138,138); }

#tooltip {
        position: absolute;
        width: auto;
        height: auto;
        padding: 10px;
        background-color: white;
        opacity: .9;
        -webkit-border-radius: 10px;
        -moz-border-radius: 10px;
        border-radius: 10px;
        -webkit-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.4);
        -moz-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.4);
        box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.4);
        pointer-events: none;
      }

      #tooltip.hidden {
        display: none;
      }

      #tooltip p {
        margin: 0;
        font-family: sans-serif;
        font-size: 10px;
        line-height: 12px;
      }

</style>
<?php
echo "<BODY>";
echo "<div id='tooltip' class='hidden'>";
echo "<p><b><span id='stations'></span></b></p><p><span
id='dateRun'></span></p><p><span id='time'>100</span></p><p><span
id='actualTime'>100</span></p></div>";

//retrieve information on origin, destination, and intermediate stations
$originQuery = "SELECT * FROM nrtiploc WHERE TIPLOCCodeID = '".$originTIPLOC."'";
$destinationQuery = "SELECT * FROM nrtiploc WHERE TIPLOCCodeID =
'".$destinationTIPLOC."'";
$viaQuery = "SELECT * FROM nrtiploc WHERE TIPLOCCodeID = '".$viaTIPLOC."'";

$originStmt = $con->prepare($originQuery);
$originStmt->execute();
$originResult = $originStmt->get_result();

$destinationStmt = $con->prepare($destinationQuery);
$destinationStmt->execute();
$destinationResult = $destinationStmt->get_result();
```

```
$viaStmt = $con->prepare($viaQuery);
$viaStmt->execute();
$viaResult = $viaStmt->get_result();

//convert the station names into first letter capitalised format
while ($originRow = $originResult->fetch_assoc()) {
        while ($destinationRow = $destinationResult->fetch_assoc()) {
            while ($viaRow = $viaResult->fetch_assoc()) {
                    $originStation = str_replace('( ', '(', ucwords(str_replace('(',
'( ', strtolower($originRow['TIPLOCDescription']))));
                    $destinationStation = str_replace('( ', '(',
ucwords(str_replace('(', '( ', strtolower($destinationRow['TIPLOCDescription']))));
                    $viaStation = str_replace('( ', '(', ucwords(str_replace('(',
'( ', strtolower($viaRow['TIPLOCDescription']))));
                    }
                }
        }

$results = array();

?>
<script type="text/javascript">

var trainResults = new Array();
</script>
<?php

$i = 0;

//select the date range we wish to extract data from
//in this case, the 19th of May is selected as the start date as the summer
schedule was implemented on this date
$dateArray = array();
$firstDate = new DateTime('2014-05-19');
$lastDate = new DateTime('2014-08-14');
$tempFirstDate = $firstDate->format('Y-m-d');

while ($row = $detailResult->fetch_assoc())
{
        $j = 0;
        //find all stops on selected schedule route
        $internalDetailQuery = "SELECT * FROM `nrscheduletrainsignals` INNER JOIN
`nrtiploc` ON (`nrscheduletrainsignals`.TIPLOCCode = `nrtiploc`.TIPLOCCodeID) WHERE
`nrscheduletrainsignals`.signalTrainID = '".$row['scheduleTrainID']."' AND
`nrscheduletrainsignals`.signalPointNo >= (SELECT signalPointNo FROM
`nrscheduletrainsignals` WHERE signalTrainID = '".$row['scheduleTrainID']."' AND
TIPLOCCode = '$originTIPLOC' LIMIT 1) AND `nrscheduletrainsignals`.signalPointNo <=
(SELECT signalPointNo FROM `nrscheduletrainsignals` WHERE signalTrainID =
'".$row['scheduleTrainID']."' AND TIPLOCCode = '$destinationTIPLOC' ORDER BY
signalPointNo DESC LIMIT 1) AND `nrscheduletrainsignals`.signalStartDate <= '2014-
05-19' AND `nrscheduletrainsignals`.signalEndDate >= '2014-05-19' AND
(`nrscheduletrainsignals`.internalArrival <> 'pass' OR
`nrscheduletrainsignals`.internalArrival IS NULL) AND `nrtiploc`.passengerStop =
'Y' ORDER BY `nrscheduletrainsignals`.signalPointNo ASC";
        $internalDetailStmt = $con->prepare($internalDetailQuery);
        $internalDetailStmt->execute();
        $internalDetailResult = $internalDetailStmt->get_result();

        //ensure that the intermediate stop selected is not before the beginning or
after the end of the journey
        //it is possible for the intermediate stop to be called on more than once
on each journey - we need to make sure we capture the correct stop
```

```
        $intermediateDetailQuery = "SELECT * FROM `nrscheduletrainsignals` INNER
JOIN `nrtiploc` ON (`nrscheduletrainsignals`.TIPLOCCode = `nrtiploc`.TIPLOCCodeID)
WHERE `nrscheduletrainsignals`.signalTrainID = '".$row['scheduleTrainID']."' AND
`nrscheduletrainsignals`.TIPLOCCode = '$viaTIPLOC' AND
`nrscheduletrainsignals`.signalPointNo >= ".$row["startNumber"]." AND
`nrscheduletrainsignals`.signalPointNo <= ".$row["endNumber"]." ORDER BY
`nrscheduletrainsignals`.signalPointNo ASC";
        $intermediateDetailStmt = $con->prepare($intermediateDetailQuery);
        $intermediateDetailStmt->execute();
        $intermediateDetailResult = $intermediateDetailStmt->get_result();

        //make sure that the train stops at the intermediate station by checking
the number of results returned
        $stopOnRoute = $intermediateDetailResult->num_rows;

        //if the train DOES stop at the intermediate stop, run the following code
        if($stopOnRoute > 0)
        {
            $i++;

            //add the train information to the next available array slot
            $results[$i] = array(
                'scheduleTrainID' => $row["scheduleTrainID"]
            );

            //retrieve the arrival and departure times for all stops on the journey
            while ($detailRow = $internalDetailResult->fetch_assoc())
            {
                $j++;

                if($j == mysqli_num_rows($internalDetailResult))
                {
                    $averageQuery = "SELECT
DATE_FORMAT(FROM_UNIXTIME(AVG(UNIX_TIMESTAMP(STR_TO_DATE(RTarrival,'%H%i')))),'%H:%
i:%s') AS 'Actual Time' FROM nrhistoricdata NRHIS WHERE NRHIS.trainID =
'".$row["scheduleTrainID"]."' AND NRHIS.trainTIPLOC =
'".$detailRow["TIPLOCCode"]."' AND NRHIS.RTarrival IS NOT NULL AND NRHIS.trainStop
>= ".$row['startNumber']." AND NRHIS.trainStop <= ".$row["endNumber"];
                }
                else
                {
                    $averageQuery = "SELECT
DATE_FORMAT(FROM_UNIXTIME(AVG(UNIX_TIMESTAMP(STR_TO_DATE(RTdeparture,'%H%i')))),'%H
:%i:%s') AS 'Actual Time' FROM nrhistoricdata NRHIS WHERE NRHIS.trainID =
'".$row["scheduleTrainID"]."' AND NRHIS.trainTIPLOC =
'".$detailRow["TIPLOCCode"]."' AND NRHIS.RTdeparture IS NOT NULL AND
NRHIS.trainStop >= ".$row['startNumber']." AND NRHIS.trainStop <=
".$row["endNumber"];
                }
                $averageStmt = $con->prepare($averageQuery);
                $averageStmt->execute();
                $averageResult = $averageStmt->get_result();
                while ($averageRow = $averageResult->fetch_assoc())
                {
                    //The visualisation breaks if there no average to calculate due
to lack of data.
                    //To keep the visualisation looking reasonable, this uses the
internal departure time
                    //as a placeholder. Not ideal, but at least there is some sense
maintained in the graph.
                    if (is_null($averageRow['Actual Time']))
                    {
                        if(substr($detailRow["internalDeparture"],-1) == "H")
```

```
                        {
                            $averageArrival =
date('H:i:s',strtotime(substr($detailRow["internalDeparture"],0,4)) + 30);
                        }
                        else
                        {
                            $averageArrival =
date('H:i:s',strtotime(substr($detailRow["internalDeparture"],0,4)));
                        }
                    }
                    else
                    {
                        $averageArrival = $averageRow['Actual Time'];
                    }
                }

                if ($detailRow["publicArrival"] <> NULL)
                {
                        $publicArrival =
date('H:i',strtotime($detailRow["publicArrival"]));
                }
                else $publicArrival = NULL;
                if ($detailRow["internalDeparture"] <> NULL)
                {
                    if(substr($detailRow["internalDeparture"],-1) == "H")
                    {
                        $publicDeparture =
date('H:i:s',strtotime(substr($detailRow["internalDeparture"],0,4)) + 30);
                    }
                    else
                    {
                        $publicDeparture =
date('H:i:s',strtotime(substr($detailRow["internalDeparture"],0,4)));
                    }
                }
                else if ($detailRow["publicDeparture"] <> NULL)
                {
                    $publicDeparture =
date('H:i:s',strtotime($detailRow["publicDeparture"]));
                }
                else $publicDeparture = NULL;

                //retrieve the days that the train ran - this is important for
being able to display the information on the graph later
                $dateArray = array();
                $daysTrainRanQuery = "SELECT runningTrains.dateRun FROM (";
                $daysTrainRanQuery .= "SELECT * FROM `nrscheduletrains` INNER JOIN
`nrhistoricdata` ON (`nrscheduletrains`.scheduleTrainID = `nrhistoricdata`.trainID
AND `nrhistoricdata`.dateRun >= `nrscheduletrains`.scheduleStartDate AND
`nrhistoricdata`.dateRun <= `nrscheduletrains`.scheduleEndDate) INNER JOIN
`nrtiploc` ON (`nrhistoricdata`.trainTIPLOC = `nrtiploc`.TIPLOCCodeID) WHERE
`nrscheduletrains`.scheduleTrainID = '".$row['scheduleTrainID']."' AND
`nrhistoricdata`.dateRun >= '".$tempFirstDate."' AND `nrhistoricdata`.dateRun <=
'".$lastDate->format('Y-m-d')."' AND `nrhistoricdata`.trainCancelReason IS NULL AND
`nrhistoricdata`.trainStop = (SELECT trainStop FROM `nrhistoricdata` WHERE trainID
= '".$row['scheduleTrainID']."' AND trainTIPLOC = '".$originTIPLOC."' LIMIT 1) AND
`nrhistoricdata`.RTdeparture IS NOT NULL ORDER BY `nrhistoricdata`.dateRun ASC) AS
runningTrains ";
                $daysTrainRanQuery .= "INNER JOIN `nrhistoricdata` ON
(runningTrains.scheduleTrainID = `nrhistoricdata`.trainID AND
`nrhistoricdata`.dateRun >= runningTrains.scheduleStartDate AND
`nrhistoricdata`.dateRun <= runningTrains.scheduleEndDate) INNER JOIN `nrtiploc` ON
(`nrhistoricdata`.trainTIPLOC = `nrtiploc`.TIPLOCCodeID) WHERE
```

```
runningTrains.scheduleTrainID = '".$row['scheduleTrainID']."' AND
`nrhistoricdata`.dateRun >= '".$tempFirstDate."' AND `nrhistoricdata`.dateRun <=
'".$lastDate->format('Y-m-d')."' AND `nrhistoricdata`.trainCancelReason IS NULL AND
`nrhistoricdata`.trainStop = (SELECT trainStop FROM `nrhistoricdata` WHERE trainID
= '".$row['scheduleTrainID']."' AND trainTIPLOC = '".$destinationTIPLOC."' ORDER BY
trainStop DESC LIMIT 1) AND `nrhistoricdata`.RTarrival IS NOT NULL AND
runningTrains.dateRun = `nrhistoricdata`.dateRun ORDER BY `nrhistoricdata`.dateRun
ASC";
                $daysTrainRanStmt = $con->prepare($daysTrainRanQuery);
                $daysTrainRanStmt->execute();
                $daysTrainRanResult = $daysTrainRanStmt->get_result();
                //push all the dates that the train ran into an array that will be
accessed later
                while ($daysTrainRanRow = $daysTrainRanResult->fetch_assoc())
                {
                        array_push($dateArray, $daysTrainRanRow["dateRun"]);
                }
                $datesFromQuery = implode("','",$dateArray);
                //using MySQL's "IN" selector is necessary, to ensure that only the
journeys between the selected dates are retrieved
                //we also need to mitigate against journeys that start and end in
the same station
                //for example, some trains from London Waterloo finish at London
Waterloo. We only want to return the origin or the destination, not both.
                $historicQuery = "SELECT * FROM (";
                $historicQuery .= "SELECT * FROM `nrscheduletrains` INNER JOIN
`nrhistoricdata` ON (`nrscheduletrains`.scheduleTrainID = `nrhistoricdata`.trainID
AND `nrhistoricdata`.dateRun >= `nrscheduletrains`.scheduleStartDate AND
`nrhistoricdata`.dateRun <= `nrscheduletrains`.scheduleEndDate) INNER JOIN
`nrtiploc` ON (`nrhistoricdata`.trainTIPLOC = `nrtiploc`.TIPLOCCodeID) WHERE
`nrscheduletrains`.scheduleTrainID = '".$row['scheduleTrainID']."' AND
`nrhistoricdata`.dateRun >= '".$tempFirstDate."' AND `nrhistoricdata`.dateRun <=
'".$lastDate->format('Y-m-d')."' AND `nrhistoricdata`.trainCancelReason IS NULL AND
`nrhistoricdata`.trainStop = (SELECT trainStop FROM `nrhistoricdata` WHERE trainID
= '".$row['scheduleTrainID']."' AND trainTIPLOC = '".$originTIPLOC."' LIMIT 1) AND
`nrhistoricdata`.RTdeparture IS NOT NULL ORDER BY `nrhistoricdata`.dateRun ASC) AS
runningTrains ";
                $historicQuery .= "INNER JOIN `nrhistoricdata` ON
(runningTrains.scheduleTrainID = `nrhistoricdata`.trainID AND
`nrhistoricdata`.dateRun >= runningTrains.scheduleStartDate AND
`nrhistoricdata`.dateRun <= runningTrains.scheduleEndDate) INNER JOIN `nrtiploc` ON
(`nrhistoricdata`.trainTIPLOC = `nrtiploc`.TIPLOCCodeID) WHERE
runningTrains.scheduleTrainID = '".$row['scheduleTrainID']."' AND
`nrhistoricdata`.dateRun >= '".$tempFirstDate."' AND `nrhistoricdata`.dateRun <=
'".$lastDate->format('Y-m-d')."' AND `nrhistoricdata`.trainCancelReason IS NULL AND
`nrhistoricdata`.trainStop = (SELECT trainStop FROM `nrhistoricdata` WHERE trainID
= '".$row['scheduleTrainID']."' AND trainTIPLOC = '".$detailRow['TIPLOCCode']."'
AND trainStop >= (SELECT trainStop FROM `nrhistoricdata` WHERE trainID =
'".$row['scheduleTrainID']."' AND trainTIPLOC = '".$originTIPLOC."' LIMIT 1) ORDER
BY trainStop ASC LIMIT 1) AND runningTrains.dateRun = `nrhistoricdata`.dateRun AND
runningTrains.dateRun IN ('".$datesFromQuery."') ORDER BY `nrhistoricdata`.dateRun
ASC";
                $historicStmt = $con->prepare($historicQuery);
                $historicStmt->execute();
                $historicResult = $historicStmt->get_result();
                $historicNumResults = mysqli_num_rows($historicResult);
                $k = 0;
                //We need to push stop data into the historic data array. But first,
we need to check if there is timing data for that stop for the day.
                if ($historicNumResults > 0)
                {
                    while ($historicRow = $historicResult->fetch_assoc()){
```

```
                            //IMPORTANT: the associative array using the date as a
key is essential, as the Javascript code will not deal with associative arrays
easily.
                            //Thus, using the date as a key here will allow us to
deal with this data more easily with the JSON string passed into the Javascript
code.
                            $historicPerformance[$historicRow["dateRun"]] = array(
                            'dateRun' => $historicRow["dateRun"],
                            'RTarrival' => $historicRow["RTarrival"],
                            'RTdeparture' => $historicRow["RTdeparture"]);
                            $k++;
                        }
                }
                //If no timing data, push a blank data set through with the date
run
                else
                {
                    foreach ($dateArray as $currentDate) {
                        $historicPerformance[$currentDate] = array(
                        'dateRun' => $currentDate,
                        'RTarrival' => NULL,
                        'RTdeparture' => NULL);
                        $k++;
                    }
                }
                //add each scheduled train to the results array, including the
historic performance array generated above
                $results[$i][$j] = array(
                    'trainStop' => $detailRow["signalPointNo"],
                    'trainTIPLOCDescription' => $detailRow["TIPLOCDescription"],
                    'longitude' => $detailRow["longitude"],
                    'latitude' => $detailRow["latitude"],
                    'passengerStop' => $detailRow["passengerStop"],
                    'trainAtPlatform' => $detailRow["internalArrival"],
                    'arrivalTime' => $publicArrival,
                    'departureTime' => $publicDeparture,
                    'averageArrival' => $averageArrival,
                    'historicPerformance' => $historicPerformance
                    );
                }
                //re-initialise the array for the next train
                $historicPerformance = array();
        }
        //to turn the array into something that Javascript can use, the results
array is turned into a JSON string
        $json = json_encode($results);
}
?>
<script type="text/javascript">
try {
    // add all stops to the array
       trainResults = JSON.parse('<?php echo empty($results) ? '{}' : $json ?>');
} catch (e) {
    if (e instanceof SyntaxError)
    {
        // report syntax error
        console.error("Cannot parse JSON", e);
    }
}

//initialise the arrays used for the train data
var data = [];
var stations = [];
```

```
//this is used for the line colours of each journey - on time trains in green,
progressively moving towards red and (for extremely late trains) purple
var color = d3.scale.linear()
    .domain([-5, 0, 0.5, 1])
    .range(["purple", "red", "yellow", "green"]);

//loop through each scheduled train to plot it on the graph
for(var i in trainResults) {

    var stops = [];
    var actual = [];

    //initialise the variables for the distance between two stations
    var distance = 0;
    var lat1 = 0;
    var lat2 = 0;
    var lon1 = 0;
    var lon2 = 0;

    var endTime = new Date("1900-01-01T00:00");

    //initialise the latitude and longitude of the origin station
    lat1 = trainResults[i][1].latitude;
    lon1 = trainResults[i][1].longitude;

    //allocate the next available array slot with the active train information;
declare a blank array for its stops
    data[i] = ({
        "number"                : trainResults[i].scheduleTrainID,
        "type"                  : "scheduled",
        "stops"                 : [],
    });

    //allocate the trainID to a variable as we need to remove it in order to
accurately calculate the number of stops
    var trainID = trainResults[i].scheduleTrainID;
    //remove the train ID from the array for the listed reason above
    delete trainResults[i].scheduleTrainID;

    var count = 0;
    //calculate the number of journeys by counting the number of unique keys in the
trainResults active train array
    for (var k in trainResults[i]) if (trainResults[i].hasOwnProperty(k)) ++count;

    var actualStops = 0;
    var justStopped = true;

    //loop through array
    for(var j = 1; j <= count; j++) {

        var signalItem = trainResults[i][j];

        var s = "1900-01-01T";
        var t = s;

        if (signalItem.departureTime === null) {
            s += signalItem.arrivalTime;
        }
        else {
            s += signalItem.departureTime;
        }
```

```
        s = new Date(s);

        //the average times at each stop must be translated into JS date format
        averageArrival = new Date("1900-01-01T" + signalItem.averageArrival);
        if(isNaN(averageArrival)) { averageArrival = new Date("1900-01-
01T01:00"); }

        if (s != null && s.getHours() < 3) s.setDate(s.getDate() + 1);

        var passengerStop = (signalItem.passengerStop === "N") ? "no" : "yes";

        lat2 = signalItem.latitude;
        lon2 = signalItem.longitude;

        //using the latitudes/longitudes of the two stations, calculate the "as the
crow flies" distance bewteen them
        distance = distance + haversine(lat1,lat2,lon1,lon2);

        //we need to capture the stations to display them on the x axis - only do
this for the first time the data is accessed
        if (i == 1) {
        stations.push({
            "trainStop"             : signalItem.trainStop,
            "stationName"           : signalItem.trainTIPLOCDescription,
            "distance"              : distance,
            "passengerStop"         : signalItem.passengerStop,
        });
        }

        //each stop is then added to the array
        stops[j-1] = ({
            "type"              : "scheduled",
            "time"              : s,
            "avgArrival"        : averageArrival,
            "passengerStop"     : signalItem.trainAtPlatform,
            "station"           : ({
                "key"               : signalItem.trainStop,
                "stationName"       : signalItem.trainTIPLOCDescription,
                "distance"          : distance,
            }),
        });

        var numDays = 0;

        //this code works its way through each journey array (hence why we used
associative arrays earlier in our PHP code -
        //using indexed arrays causes problems when trains do not have the same
number of journeys or even the same dates!)
        Object.keys(signalItem.historicPerformance).forEach(function (key) {
            var t = "1900-01-01T";
            var todayData = [];

            if (signalItem.historicPerformance[key].RTdeparture === null) {
                if (signalItem.historicPerformance[key].RTarrival === null) {
                    t = null;
                }
                else {
                    t +=
signalItem.historicPerformance[key].RTarrival.substring(0,2) + ":";
                    t +=
signalItem.historicPerformance[key].RTarrival.substring(2,4);
                }
            }
```

```
            else
            {
                t += signalItem.historicPerformance[key].RTdeparture.substring(0,2)
+ ":";
                t +=
signalItem.historicPerformance[key].RTdeparture.substring(2,4);
            }
            if (t !== null)
            {
                t = new Date(t);
                //the endTime is used to generate the axis on the graph. We want to
find the latest time we need to show on the graph.
                //this also takes into account trains that begin before midnight
and run past midnight. It is not likely the trains will
                //run past 3am... future work would incorporate how to handle
overnight or 24+ hour trains
                if (t > endTime) endTime = t;
                if (t != null && t.getHours() < 3) t.setDate(t.getDate() + 1);
                todayData = ({
                    "number"            : trainID,
                    "type"               : "actual",
                    "dateRun"            :
signalItem.historicPerformance[key].dateRun,

                    "scheduleTime"       : s,
                    "time"               : t,
                    "passengerStop"      : signalItem.trainAtPlatform,
                    "station"            : ({
                        "key"                : signalItem.trainStop,
                        "stationName"        : signalItem.trainTIPLOCDescription,
                        "distance"           : distance,
                    }),
                });
                //initialise the array for the new journey data
                if (!actual[numDays]) actual[numDays] = [];
                //then allocate the data for this date into the array
                actual[numDays][actualStops] = todayData;
                //justStopped indicates whether the train has called at a station.
This is to remove any calling points that were not serviced by this train
                justStopped = true;
            }
            else
            {
                justStopped = false;
            }
            numDays++;

        if(justStopped === true)
        {
            actualStops++;
        }

        //the second stop in our pair, becomes the first stop of the next pair of
stations
        lat1 = signalItem.latitude;
        lon1 = signalItem.longitude;


        });
    }
    //push the stops and actual data into two seperate parts of the array, one for
the scheduled route, and one for historic performance
    data[i]["stops"] = stops;
```

```
    data[i]["actual"] = actual;

}

var numStops = data[1]["stops"].length - 1;
var trains = 0;
var act = 0;
var performanceData = [];

//this next section ensures that the performance data is pushed into the right
section of the array
Object.keys(data).forEach(function (key) {
    var numTrains = data[key]["actual"].length;
    for (var numServices = 0; numServices < numTrains; numServices++) {
        if (!performanceData[act]) performanceData[act] = [];
        var thisTrain = data[key].actual[numServices];
            performanceData[act] = ({
            "trainID"           : thisTrain[numServices].number,
            "dateRun"           : thisTrain[numServices].dateRun,
            "type"              : "actual",
            "stops"             : thisTrain,
        });
        act++;
    };
});

//this next section re-indexes the array, to remove any spaces that have been
inserted due to missing data, cancelled journeys, etc.
//this is to ensure that the D3 code below works with a well-ordered array of
historic data
var servs = 0;
for (var k in performanceData) if (performanceData.hasOwnProperty(k)) ++servs;
for (var int = 0; int < servs; int++)
{
    var numCalls = 0;
    var stops = [];
    var testStopsArray = reindex_array_keys(performanceData[int].stops);
    var testArray = reindex_array_keys(performanceData[int]);
    performanceData[int] = ({
    "trainID"           : testArray[0],
    "dateRun"           : testArray[1],
    "type"              : testArray[2],
    "stops"             : testStopsArray,
    });
}

var startTime = data[1]["stops"][1]["time"];

var formatTime = d3.time.format("%I:%M%p");

//the graph area is defined here
var margin = {top: 80, right: 30, bottom: 20, left: 200},
    width = 1200 - margin.left - margin.right,
    height = 600 - margin.top - margin.bottom;

//the scales are rounded to the quarter hour before the first train's departure
time and the final train's scheduled arrival time
var x = d3.time.scale()
    .domain([parseTime(roundTimeQuarterHour(startTime, "down")),
parseTime(roundTimeQuarterHour(endTime, "up"))])
    .range([0, width]);

var y = d3.scale.linear()
```

```
    .range([0, height]);

var xAxis = d3.svg.axis()
    .scale(x)
    .ticks(8)
    .tickFormat(formatTime);

var line = d3.svg.line()
    .x(function(d) { return x(d.time); })
    .y(function(d) { return y(d.station.distance); });

var svg = d3.select("body").append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
  .append("g")
    .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

svg.append("defs").append("clipPath")
    .attr("id", "clip")
  .append("rect")
    .attr("y", -margin.top)
    .attr("width", width)
    .attr("height", height + margin.top + margin.bottom);

y.domain(d3.extent(stations, function(d) { return d.distance; }));

var station = svg.append("g")
    .attr("class", "station")
  .selectAll("g")
    .data(stations)
  .enter().append("g")
    .attr("transform", function(d) { return "translate(0," + y(d.distance) +
")"; });

station.append("text")
    .attr("x", -6)
    .attr("dy", ".35em")
    .style("font-style", function(d) { return d.passengerStop === "Y"? "normal" :
"italic"; })
    .style("font-weight", function(d) { return d.passengerStop === "Y"? "bold" :
"normal"; })
    .text(function(d) { return d.stationName; });

station.append("line")
    .attr("x2", width);

svg.append("g")
    .attr("class", "x top axis")
    .call(xAxis.orient("top"));

svg.append("g")
    .attr("class", "x bottom axis")
    .attr("transform", "translate(0," + height + ")")
    .call(xAxis.orient("bottom"));

//the trainPerf variable contains data on the path of the historic journey
var trainPerf = svg.append("g")
    .attr("class", "train")
    .attr("clip-path", "url(#clip)")
  .selectAll("g")
    .data(performanceData)
    .enter().append("g")
    .attr("class", function(d) { return d.type; });
```

```
trainPerf.append("path")
    .attr("d", function(d) { return line(d.stops); })
    //the minsLate variable determines the color of the line, by subtracting the
scheduled arrival time from the actual arrival time at the destination
    //originally, this was done at each stop; changing the code to use path rather
than line meant that each segment could not be coloured
    .style("stroke", function(d) { var minsLate = Math.max(0,
(d.stops[d.stops.length - 1].time - d.stops[d.stops.length - 1].scheduleTime) /
60000); return color(1 - (minsLate/5)); })
    .style("stroke-width", function (d) { return 1.5; })
    //using a more opaque line helps identify patterns in the data; the more
vibrant a colour, the more that this time has been the actual arrival time
    .style("stroke-opacity", function (d) { return 0.3; })
    .style("fill", "none")
    //hovering over the line makes the line identifiable, as all other lines are
diminished. Moving the mouse returns the line to normal.
    .on("mouseover", function(d) {
        d3.selectAll("path")
        .style("stroke-opacity", function (d) { return 0.1; });
        d3.select(this)
        .style("stroke-width", function (d) { return 5; })
        .style("stroke-opacity", function (d) { return 1; });
     })
    .on("mouseout", function(d) {
        d3.selectAll("path")
        .style("stroke-opacity", function (d) { return 0.3; });
        d3.select(this)
         .style("stroke-width", function (d) { return 1.5; })
         .style("stroke-opacity", function (d) { return 0.3; });
     });

//this code draws the station on the path using a circle. Hovering over the circle
gives information to the user about the train's journey
//and the average and scheduled journeys
trainPerf.selectAll("circle")
      .data(function(d) { return d.stops; })
    .enter().append("circle")
      .attr("transform", function(d) { return "translate(" + x(d.time) + "," +
y(d.station.distance) + ")"; })
      .attr("r", function(d) { return d.passengerStop === "pass"? 0 : 3 })
       .on("mouseover", function(d) {
                  var xPosition = x(d.time)+ margin.left + margin.right;
                  var yPosition = y(d.station.distance) + (margin.top / 2);
        var colour = "green";
        d3.select(this).style("fill", colour);
        d3.select("#tooltip")
            .style("left", xPosition + "px")
            .style("top", yPosition + "px")
            .select("#stations")
            .text(d.station.stationName);
      d3.select("#tooltip")
            .select("#dateRun")
            .text(d.dateRun);
      d3.select("#tooltip")
            .select("#time")
            .text("Train ID: " + d.number);
      d3.select("#tooltip")
            .select("#actualTime")
            .text("Scheduled Time: " + formatTime(d.scheduleTime) + " - Actual
Time: " + formatTime(d.time));
      d3.select("#tooltip").classed("hidden", false);
            })
```

```
                .on("mouseout", function() {
                    d3.select(this).style("fill", "black");
                    d3.select("#tooltip").classed("hidden", true);
                          });

//this draws the scheduled journey
var train = svg.append("g")
    .attr("class", "train")
    .attr("clip-path", "url(#clip)")
  .selectAll("g")
    .data(data.filter(function(d) { return d.type; }))
  .enter().append("g")
    .attr("class", function(d) { return d.type; });

train.selectAll("line")
      .data(function(d) { return d.stops.slice(1).map(function(b, i) { return
[d.stops[i], b]; }); })
    .enter().append("line")
      .attr("class", "scheduled")
      .attr("x1", function(d) { return x(d[0].time); })
      .attr("x2", function(d) { return x(d[1].time); })
      .attr("y1", function(d) { return y(d[0].station.distance); })
      .attr("y2", function(d) { return y(d[1].station.distance); })
      .style("stroke", function(d) { return "black"; })
      .style("stroke-width", function (d) { return 3; });

train.selectAll("circle")
      .data(function(d) { return d.stops; })
    .enter().append("circle")
      .attr("transform", function(d) { return "translate(" + x(d.time) + "," +
y(d.station.distance) + ")"; })
      .attr("r", function(d) { return d.passengerStop === "pass"? 0 : 3 })
       .on("mouseover", function(d) {
                  var xPosition = x(d.time)+ margin.left + margin.right;
                  var yPosition = y(d.station.distance) + (margin.top / 2);
        var colour = "black";
        d3.select(this).style("fill", colour);
        d3.select("#tooltip")
            .style("left", xPosition + "px")
            .style("top", yPosition + "px")
            .select("#stations")
            .text(d.station.stationName);
      d3.select("#tooltip")
            .select("#dateRun")
            .text("");
      d3.select("#tooltip")
            .select("#time")
            .text("Scheduled Time: " + formatTime(d.time));
      d3.select("#tooltip")
            .select("#actualTime")
            .text("Average Time: " + formatTime(d.avgArrival));
      d3.select("#tooltip").classed("hidden", false);
            })
                .on("mouseout", function() {
                    d3.select(this).style("fill", "black");
                    d3.select("#tooltip").classed("hidden", true);
                          });

//this draws the average journey
//the three code segments are very similar, but must be called seperately in order
for the correct layering to take place
var avgTrain = svg.append("g")
    .attr("class", "train")
```

```
    .attr("clip-path", "url(#clip)")
  .selectAll("g")
    .data(data.filter(function(d) { return d.type; }))
  .enter().append("g")
    .attr("class", function(d) { return d.type; });

avgTrain.selectAll("line")
      .data(function(d) { return d.stops.slice(1).map(function(b, i) { return
[d.stops[i], b]; }); })
    .enter().append("line")
      .attr("class", "scheduled")
      .attr("x1", function(d) { return x(d[0].avgArrival); })
      .attr("x2", function(d) { return x(d[1].avgArrival); })
      .attr("y1", function(d) { return y(d[0].station.distance); })
      .attr("y2", function(d) { return y(d[1].station.distance); })
      .style("stroke", function(d) { return "blue"; })
      .style("stroke-width", function (d) { return 3; });

avgTrain.selectAll("circle")
      .data(function(d) { return d.stops; })
    .enter().append("circle")
      .attr("transform", function(d) { return "translate(" + x(d.avgArrival) + ","
+ y(d.station.distance) + ")"; })
      .attr("r", function(d) { return d.passengerStop === "pass"? 0 : 3 })
       .on("mouseover", function(d) {
                  var xPosition = x(d.avgArrival)+ margin.left + margin.right;
                  var yPosition = y(d.station.distance) + (margin.top / 2);
        var colour = "grey";
        d3.select(this).style("fill", colour);
        d3.select("#tooltip")
            .style("left", xPosition + "px")
            .style("top", yPosition + "px")
            .select("#stations")
            .text(d.station.stationName);
      d3.select("#tooltip")
            .select("#dateRun")
            .text("");
      d3.select("#tooltip")
            .select("#time")
            .text("Scheduled Time: " + formatTime(d.time));
      d3.select("#tooltip")
            .select("#actualTime")
            .text("Average Time: " + formatTime(d.avgArrival));
      d3.select("#tooltip").classed("hidden", false);
            })
                .on("mouseout", function() {
                    d3.select(this).style("fill", "black");
                    d3.select("#tooltip").classed("hidden", true);
                          });


svg.append("text")
        .attr("x", (width / 2))
        .attr("y", 0 - (margin.top / 2))
        .attr("text-anchor", "middle")
        .style("font-size", "16px")
        .style("text-decoration", "underline")
        .text("Train Performance vs Schedule <?php echo "from ".$originStation." to
".$destinationStation." (via ".$viaStation.")"; ?>");

//this is a quick legend added to the graph to explain to users what the black and
blue lines mean
svg.append("text")
```

```
        .attr("x", width - (width / 8))
        .attr("y", 20)
        .attr("text-anchor", "middle")
        .style("font-size", "12px")
        .attr("fill", "black")
        .text("Black line shows scheduled journey");

svg.append("text")
        .attr("x", width - (width / 8))
        .attr("y", (margin.top / 2))
        .attr("text-anchor", "middle")
        .style("font-size", "12px")
        .attr("fill", "blue")
        .text("Blue line shows average journey");


function parseTime(s) {
    var t = formatTime.parse(s);
    if (t != null && t.getHours() < 3) t.setDate(t.getDate() + 1);
    return t;
}

function toRadians(x) {
   return x * Math.PI / 180;
}

//this function returns the straight line distance between two points, given their
geographic co-ordinates
function haversine(lat1, lat2, lon1, lon2) {
var R = 6371; // km
var R1 = toRadians(lat1);
var R2 = toRadians(lat2);
var S1 = toRadians(lat2 - lat1);
var S2 = toRadians(lon2 - lon1);

var a = Math.sin(S1/2) * Math.sin(S1/2) +
        Math.cos(R1) * Math.cos(R2) *
        Math.sin(S2/2) * Math.sin(S2/2);
var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

var d = R * c;

return d;
}

//to fully show the train journeys on the graph, this function will round the time
up or down to the nearest 15 minutes,
//allowing the graph to show each journey without clipping data
function roundTimeQuarterHour(time, direction) {
    var timeToReturn = new Date(time);

    timeToReturn.setMilliseconds(Math.round(timeToReturn.getMilliseconds() / 1000)
* 1000);
    timeToReturn.setSeconds(Math.round(timeToReturn.getSeconds() / 60) * 60);
    if (direction === "down") {
        timeToReturn.setMinutes((Math.round(timeToReturn.getMinutes() / 15) * 15) -
15);
    }
    else
    {
        timeToReturn.setMinutes((Math.round(timeToReturn.getMinutes() / 15) * 15) +
15);
    }
```

```
    return formatAMPM(timeToReturn);
}

//this function converts the JS 24 hour time into 12 hour time for the purposes of
displaying on the graph axis and tool tips
function formatAMPM(date) {
  var hours = date.getHours();
  var minutes = date.getMinutes();
  var ampm = hours >= 12 ? 'PM' : 'AM';
  hours = hours % 12;
  hours = hours ? hours : 12; // the hour '0' should be '12'
  minutes = minutes < 10 ? '0'+minutes : minutes;
  var strTime = hours + ':' + minutes + '' + ampm;
  return strTime;
}

//this function re-indexes any array that has had gaps inserted in due to cancelled
trains, non-running services, etc.
//necessary for using the D3 array code for displaying trains
function reindex_array_keys(array, start){
    var temp = [];
    start = typeof start == 'undefined' ? 0 : start;
    start = typeof start != 'number' ? 0 : start;
    for(i in array){
        temp[start++] = array[i];
    }
    return temp;
}
</script>
<?php
echo "</BODY></HTML>";
?>
```

APPENDIX 3: DATA EXAMPLES

SCHEDULE

STATION DATA


```
{"TiplocV1":

        {"transaction_type":"Create",
        "tiploc_code":"NWCOURT",
        "nalco":"958900",
        "stanox":"83476",
        "crs_code":"NCO",
        "description":"NEWCOURT",
        "tps_description":"NEWCOURT"}

}
```


SCHEDULED TRAIN DATA


```
{"JsonScheduleV1":

        {"CIF_bank_holiday_running":null,
        "CIF_stp_indicator":"P",
        "CIF_train_uid":"W07835",
        "applicable_timetable":"Y",
        "atoc_code":"SE",
        "new_schedule_segment":

                {"traction_class":"","uic_code":""},

        "schedule_days_runs":"0000001",
        "schedule_end_date":"2014-12-07",
        "schedule_segment":

                {"signalling_id":"2D45",
                "CIF_train_category":"OO",
                "CIF_headcode":"",
                "CIF_course_indicator":1,
                "CIF_train_service_code":"24607006",
                "CIF_business_sector":"??",
                "CIF_power_type":"EMU",
                "CIF_timing_load":null,
                "CIF_speed":"090",
                "CIF_operating_characteristics":null,
                "CIF_train_class":"S",
                "CIF_sleepers":null,
                "CIF_reservations":null,
                "CIF_connection_indicator":null,
                "CIF_catering_code":null,
                "CIF_service_branding":"",
                "schedule_location":[

                        {"location_type":"LO",
                        "record_identity":"LO",
                        "tiploc_code":"STNGBRN",
```

"tiploc_instance":null,
"departure":"1512",
"public_departure":"1512",
"platform":"3",
"line":null,
"engineering_allowance":null,
"pathing_allowance":null,
"performance_allowance":null},

{"location_type":"LI",
"record_identity":"LI",
"tiploc_code":"EJSTNGB",
"tiploc_instance":null,
"arrival":null,
"departure":null,
"pass":"1514",
"public_arrival":null,
"public_departure":null,
"platform":null,
"line":null,
"path":null,
"engineering_allowance":null,
"pathing_allowance":null,
"performance_allowance":null},

{"location_type":"LI",
"record_identity":"LI",
"tiploc_code":"KMSLY",
"tiploc_instance":null,
"arrival":"1516H",
"departure":"1517",
"pass":null,
"public_arrival":"1517",
"public_departure":"1517",
"platform":null,
"line":null,
"path":null,
"engineering_allowance":null,
"pathing_allowance":null,
"performance_allowance":null},

{"location_type":"LI",
"record_identity":"LI",
"tiploc_code":"SWALE",
"tiploc_instance":null,
"arrival":"1520",
"departure":"1520H",
"pass":null,
"public_arrival":"1520",
"public_departure":"1520",
"platform":null,
"line":null,
"path":null,
"engineering_allowance":null,

"pathing_allowance":null,
"performance_allowance":null},

{"location_type":"LI",
"record_identity":"LI",
"tiploc_code":"QUENBRO",
"tiploc_instance":null,
"arrival":"1524",
"departure":"1524H",
"pass":null,
"public_arrival":"1524",
"public_departure":"1524",
"platform":null,
"line":null,
"path":null,
"engineering_allowance":null,
"pathing_allowance":null,
"performance_allowance":null},

{"location_type":"LT",
"record_identity":"LT",
"tiploc_code":"SHRNSOS",
"tiploc_instance":null,
"arrival":"1529",
"public_arrival":"1529",
"platform":"1"
,"path":null}]},

"schedule_start_date":"2014-05-18",
"train_status":"P",
"transaction_type":"Create"}}

PERFORMANCE

{"serviceUid":"W07835",
"runDate":"2014-09-21",
"serviceType":"train",
"isPassenger":true,
"trainIdentity":"2D45",
"powerType":"EMU",
"trainClass":"S",
"atocCode":"SE",
"atocName":"Southeastern",
"performanceMonitored":true,
"origin":[{

"tiploc":"STNGBRN",
"description":"Sittingbourne",
"workingTime":"151200",
"publicTime":"1512"}],

"destination":[{

"tiploc":"SHRNSOS",
"description":"Sheerness-on-Sea",

```
        "workingTime":"152900",
        "publicTime":"1529"}],

"locations":[

        {"realtimeActivated":true,
        "tiploc":"STNGBRN",
        "crs":"SIT",
        "description":"Sittingbourne",
        "gbttBookedDeparture":"1512",
        "origin":[{

                "tiploc":"STNGBRN",
                "description":"Sittingbourne",
                "workingTime":"151200",
                "publicTime":"1512"}],

        "destination":[{

                "tiploc":"SHRNSOS",
                "description":"Sheerness-on-Sea",
                "workingTime":"152900",
                "publicTime":"1529"}],

        "isCall":true,
        "isPublicCall":true,
        "realtimeDeparture":"1512",
        "realtimeDepartureActual":true,
        "platform":"3",
        "platformConfirmed":true,
        "platformChanged":false,
        "displayAs":"ORIGIN"},

        {"realtimeActivated":true,
        "tiploc":"KMSLY",
        "crs":"KML",
        "description":"Kemsley",
        "gbttBookedArrival":"1517",
        "gbttBookedDeparture":"1517",
        "origin":[{

                "tiploc":"STNGBRN",
                "description":"Sittingbourne",
                "workingTime":"151200",
                "publicTime":"1512"}],

        "destination":[{

                "tiploc":"SHRNSOS",
                "description":"Sheerness-on-Sea",
                "workingTime":"152900",
                "publicTime":"1529"}],

        "isCall":true,
        "isPublicCall":true,
        "realtimeArrival":"1516",
        "realtimeArrivalActual":true,
```

"realtimeDeparture":"1516",
"realtimeDepartureActual":true,
"realtimeGbttArrivalLateness":null,
"realtimeGbttDepartureLateness":null,
"platform":"2",
"platformConfirmed":true,
"platformChanged":false,
"displayAs":"CALL"},

{"realtimeActivated":true,
"tiploc":"SWALE",
"crs":"SWL",
"description":"Swale",
"gbttBookedArrival":"1520",
"gbttBookedDeparture":"1520",
"origin":[{

"tiploc":"STNGBRN",
"description":"Sittingbourne",
"workingTime":"151200",
"publicTime":"1512"}],

"destination":[{

"tiploc":"SHRNSOS",
"description":"Sheerness-on-Sea",
"workingTime":"152900",
"publicTime":"1529"}],

"isCall":true,
"isPublicCall":true,
"realtimeArrival":"1519",
"realtimeArrivalActual":true,
"realtimeDeparture":"1520",
"realtimeDepartureActual":true,
"realtimeGbttArrivalLateness":null,
"platform":"1",
"platformConfirmed":true,
"platformChanged":false,
"displayAs":"CALL"},

{"realtimeActivated":true,
"tiploc":"QUENBRO",
"crs":"QBR",
"description":"Queenborough",
"gbttBookedArrival":"1524",
"gbttBookedDeparture":"1524",
"origin":[{

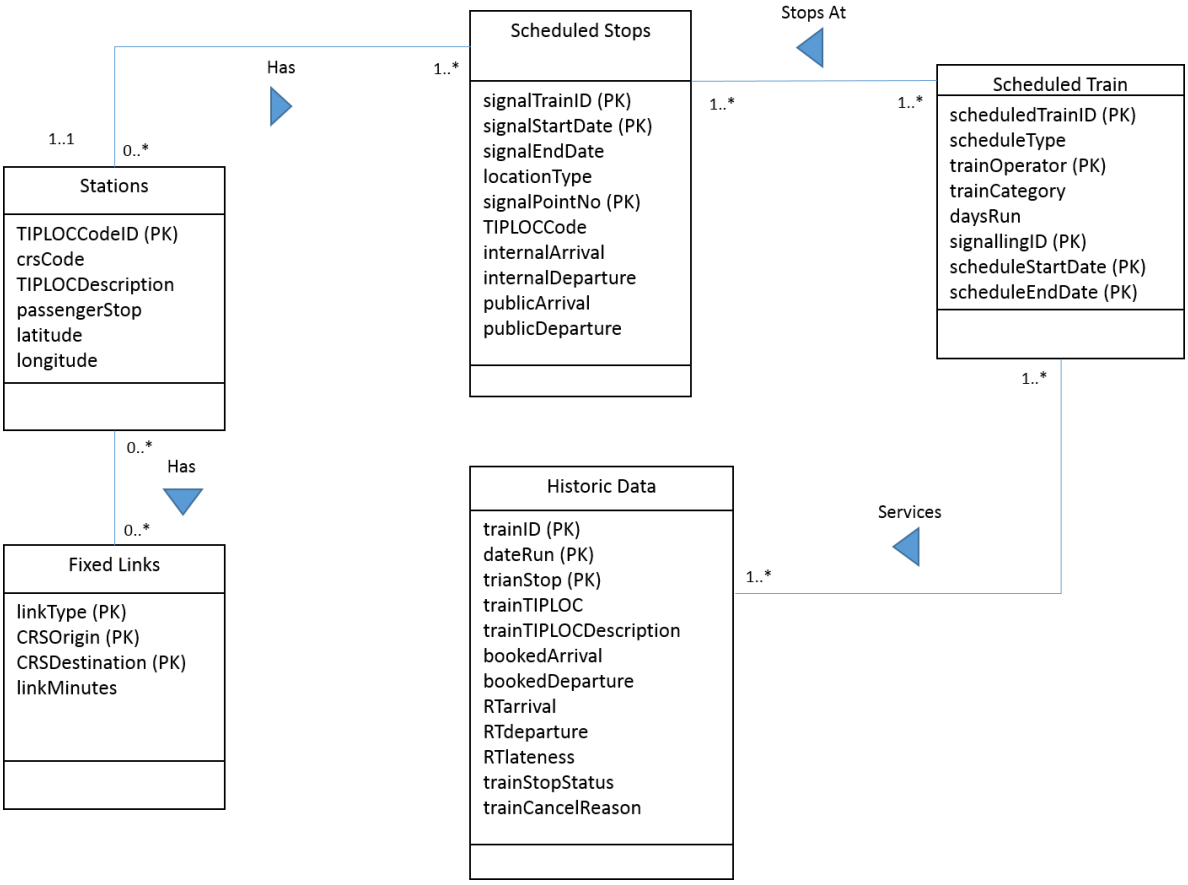
"tiploc":"STNGBRN",
"description":"Sittingbourne",
"workingTime":"151200",
"publicTime":"1512"}],

"destination":[{

```
                "tiploc":"SHRNSOS",
                "description":"Sheerness-on-Sea",
                "workingTime":"152900",
                "publicTime":"1529"}],

        "isCall":true,
        "isPublicCall":true,
        "realtimeArrival":"1524",
        "realtimeArrivalActual":true,
        "realtimeDeparture":"1524",
        "realtimeDepartureActual":true,
        "platform":"2",
        "platformConfirmed":true,
        "platformChanged":false,
        "displayAs":"CALL"},

        {"realtimeActivated":true,
        "tiploc":"SHRNSOS",
        "crs":"SSS",
        "description":"Sheerness-on-Sea",
        "gbttBookedArrival":"1529",
        “origin":[{

                "tiploc":"STNGBRN",
                "description":"Sittingbourne",
                "workingTime":"151200",
                "publicTime":"1512"}],

        "destination":[{

                "tiploc":"SHRNSOS",
                "description":"Sheerness-on-Sea",
                "workingTime":"152900",
                "publicTime":"1529"}],

        "isCall":true,
        "isPublicCall":true,
        "realtimeArrival":"1529",
        "realtimeArrivalActual":true,
        "platform":"1",
        "platformConfirmed":true,
        "platformChanged":false,
        "path":"M",
        "pathConfirmed":true,
        "displayAs":"DESTINATION"}],

"realtimeActivated":true,"runningIdentity":"2D45"}
```

APPENDIX 4: DATABASE DESIGN

ENTITY-RELATIONSHIP DIAGRAM



RELATIONAL MODEL

NRTIPLOC (**TIPLOCCodeID**, crsCode, TIPLOCDescription, passengerStop, longitude, latitude)

NRSCHEDULETRAINS (**scheduleTrainID**, scheduleType, **trainOperator**, trainCategory, daysRun, **signallingID**, **scheduleStartDate**, **scheduleEndDate**)

NRSCHEDULETRAINSIGNALS (**signalTrainID**, **signalStartDate**, signalEndDate, locationType, **signalPointNo**, TIPLOCCode, internalArrival, internalDeparture, publicArrival, publicDeparture)

NRHISTORICDATA (**trainID**, **dateRun**, **trainStop**, trainTIPLOC, bookedArrival, trainTIPLOCDescription, bookedDeparture, RTarrival, RTdeparture, RTlateness, trainStopStatus, trainCancelReason)

FOREIGN KEY trainTIPLOC REFERENCES NRTIPLOC (TIPLOCCodeID)

NRFIXEDLINKS (**linkType**, **crsOrigin**, **crsDestination**, linkMinutes)

FOREIGN KEY crsOrigin REFERENCES NRTIPLOC (TIPLOCCodeID)

FOREIGN KEY crsDestination REFERENCES NRTIPLOC (TIPLOCCodeID)

APPENDIX 5: EVOLUTION OF VISUALISATIONS

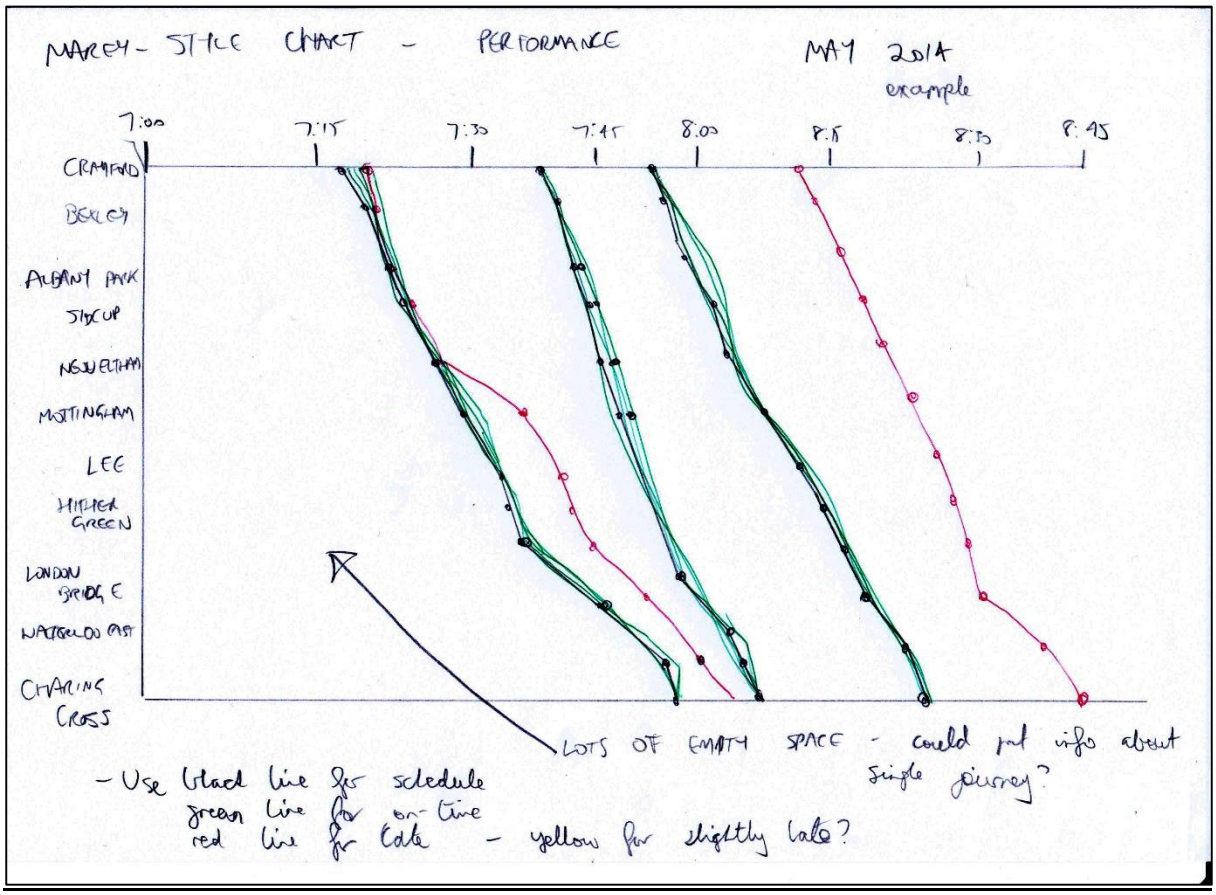


Figure 7.1. Paper prototype sketch of a Marey chart for multiple schedules and journeys

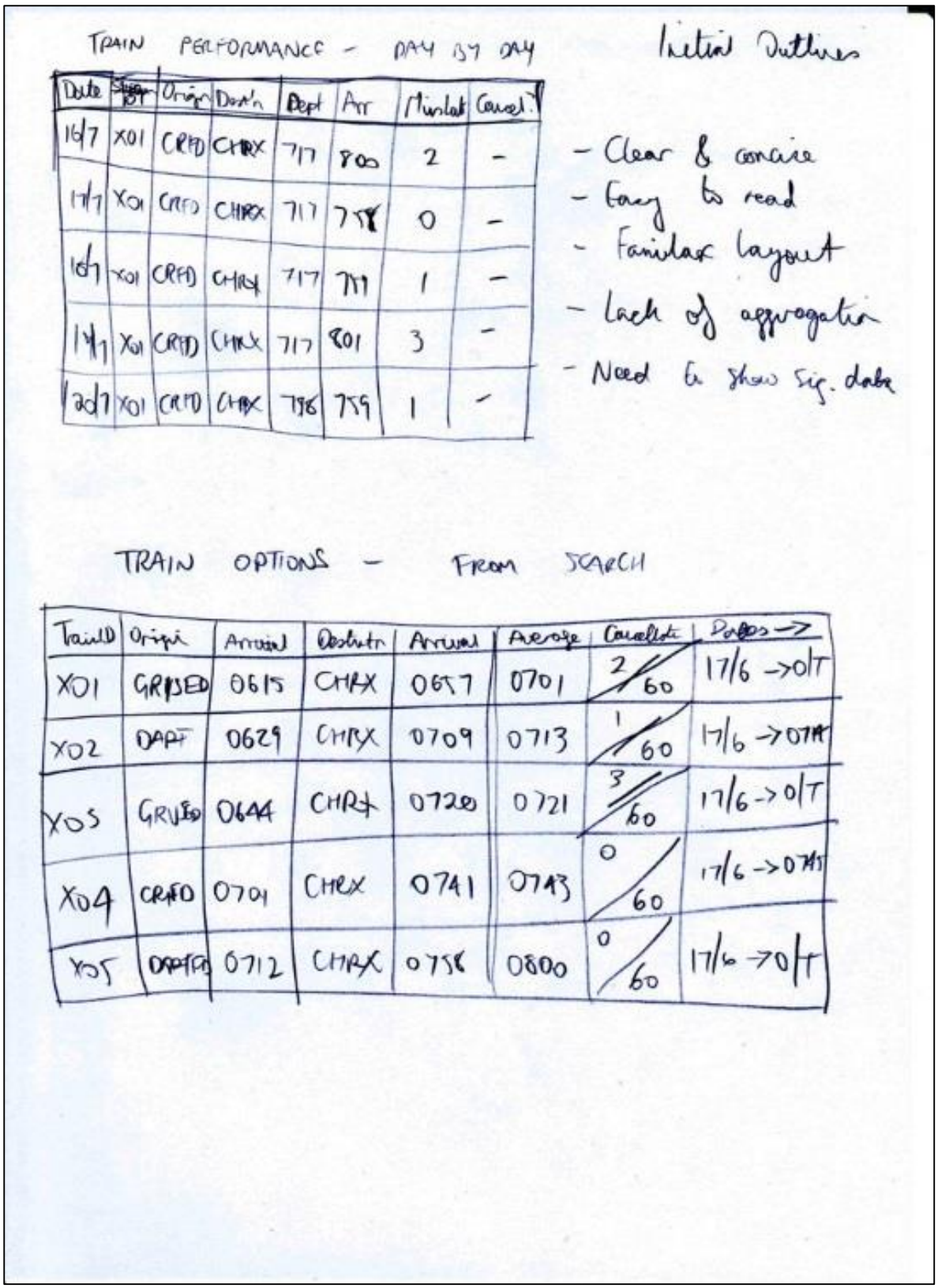


Figure 7.2. Paper prototype sketch of tabular visualisations, showing results of single journeys, and search results for multiple journeys.

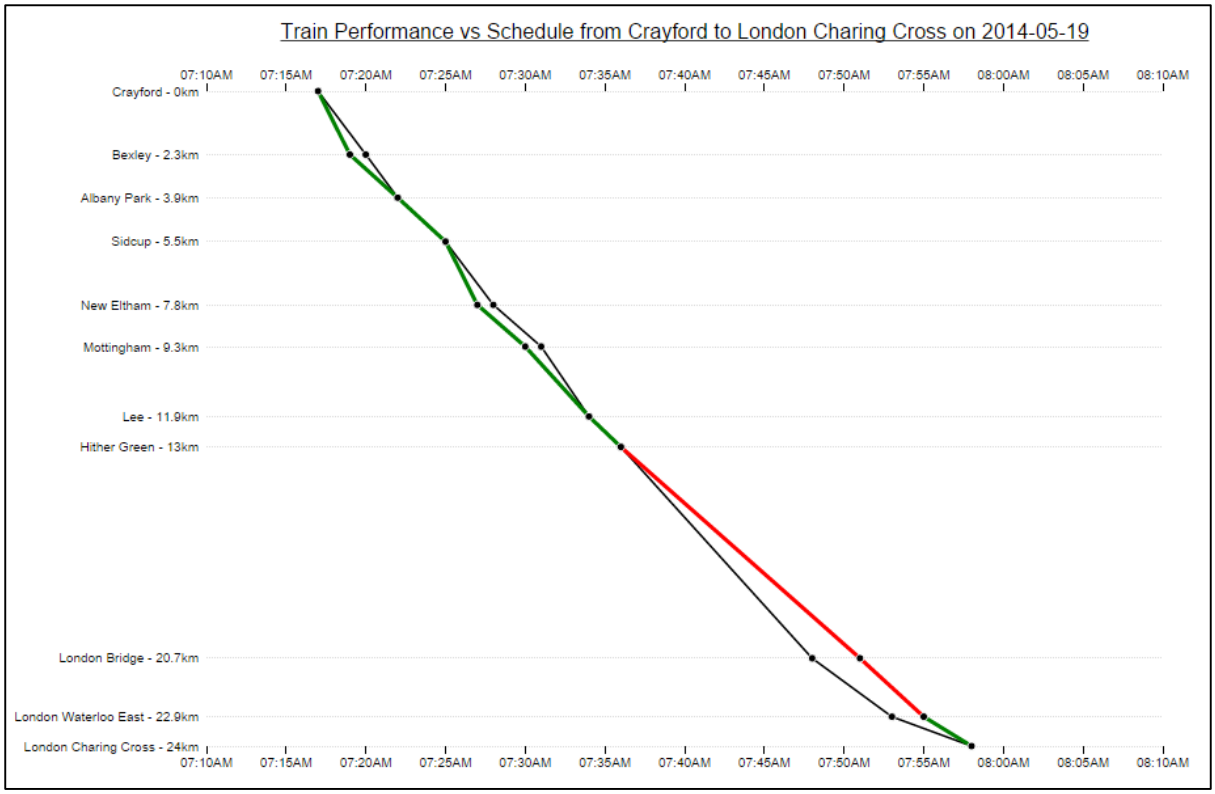


Figure 7.3. Marey chart for a single service and journey conducted on 19/5/14

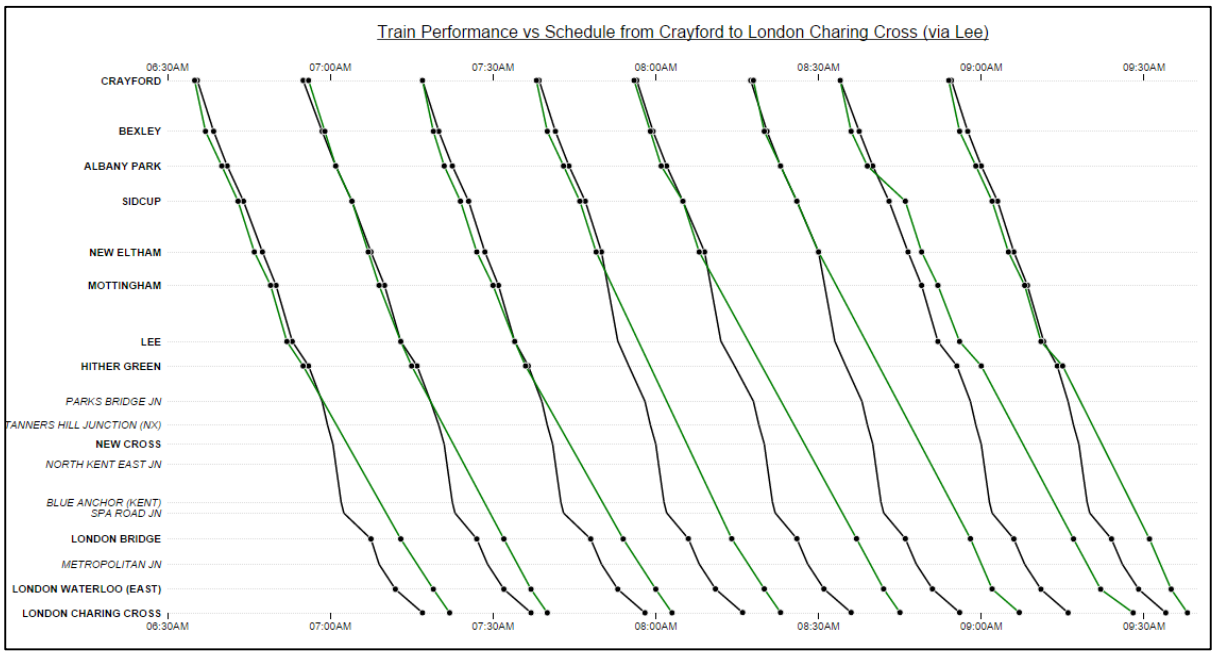


Figure 7.4. Marey chart for multiple services and journeys conducted on 19/5/14

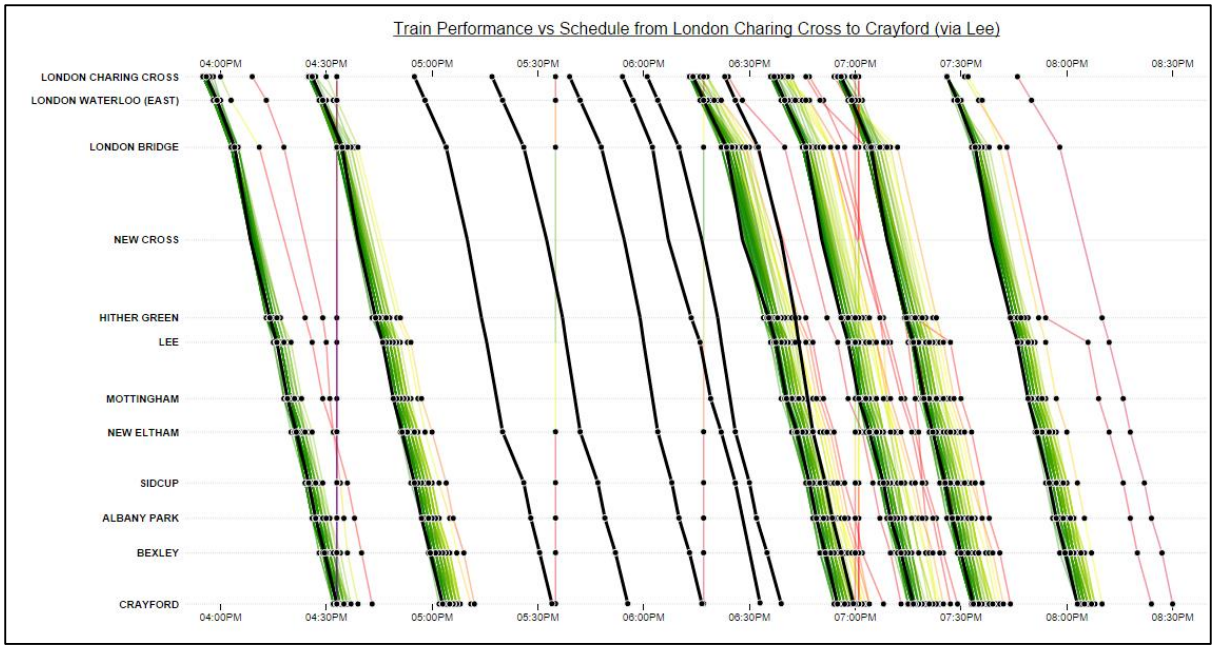


Figure 7.5. Marey chart for multiple services (erroneous data due to programming error), all journeys conducted from 19/5/14 to 24/8/14

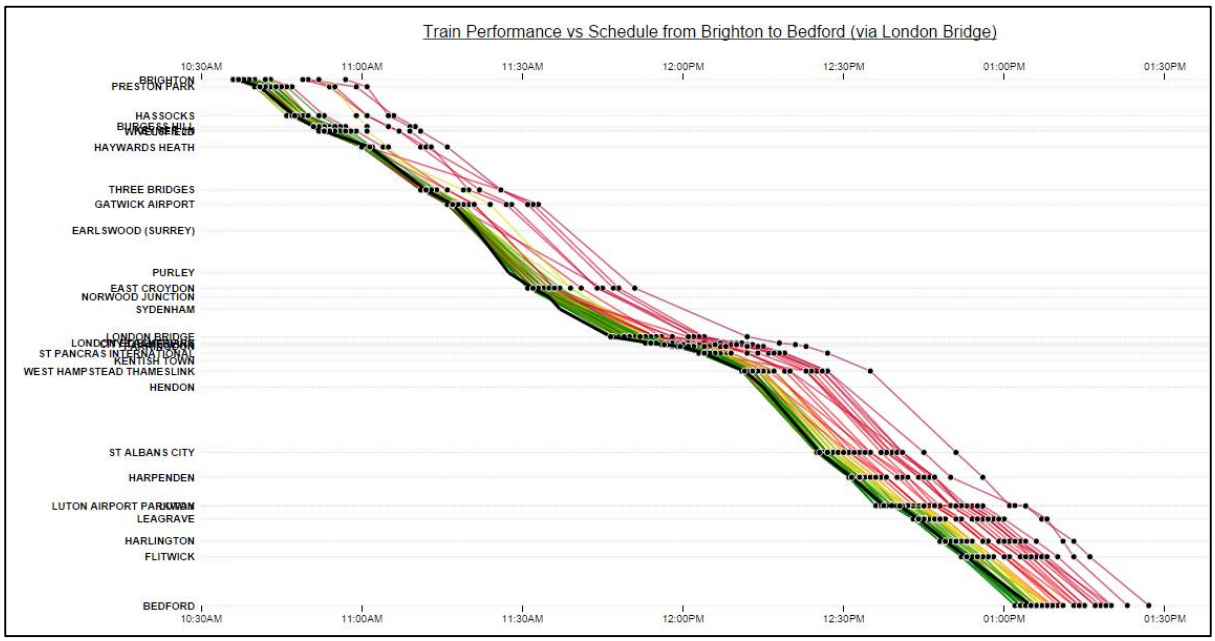


Figure 7.6. Marey chart for one service, all journeys conducted from 19/5/14 to 24/8/14

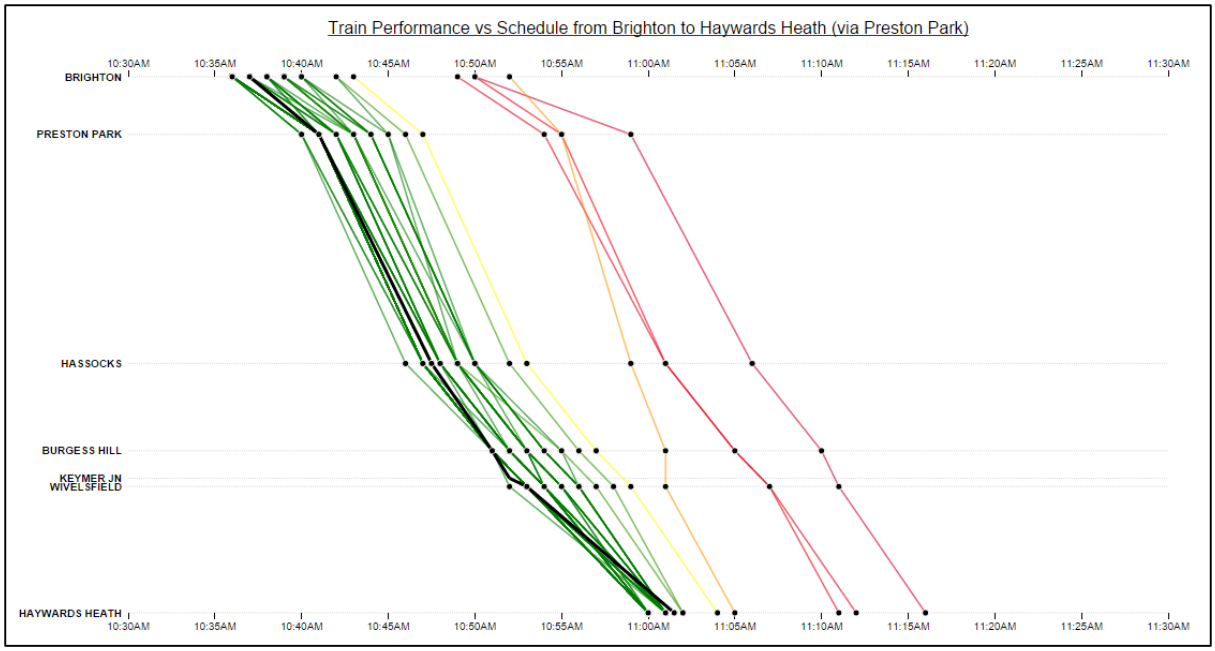


Figure 7.7. Marey chart for one service (segment of data taken from Figure 7.2.), all journeys conducted from 19/5/14 to 24/8/14

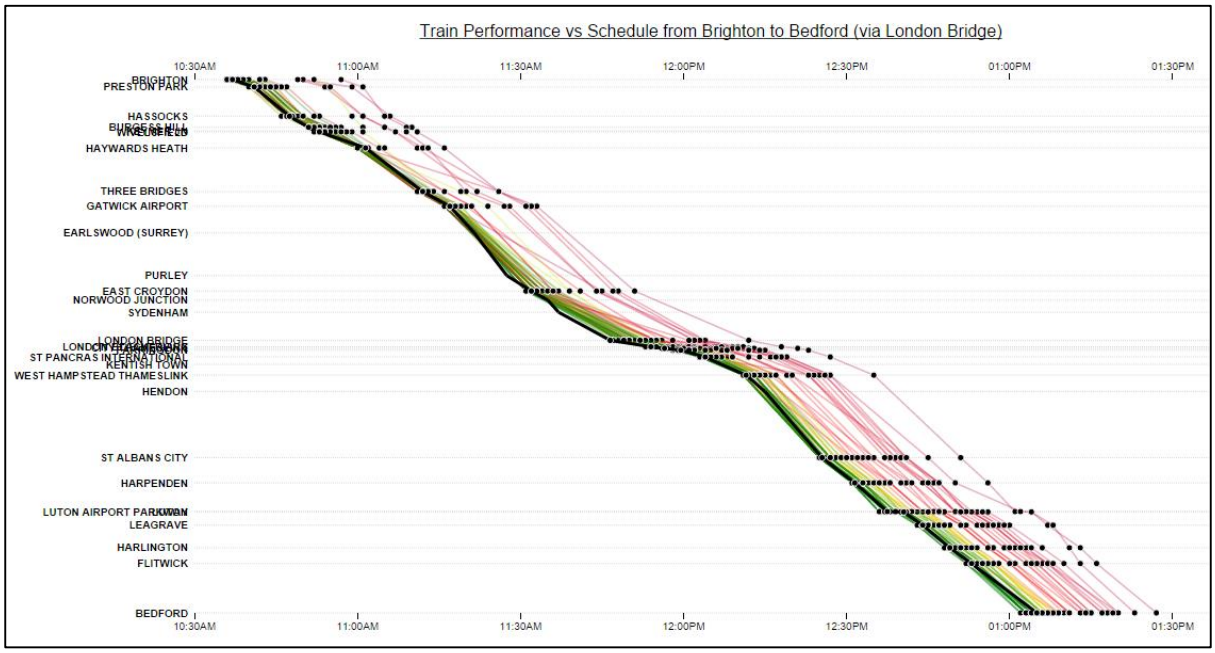


Figure 7.8. Marey chart for one service (same data as Figure 7.2.) with journey opacity set at 30%, all journeys conducted from 19/5/14 to 24/8/14

Daily train performance - trains calling at Crayford between 0700 and 2000 (weekday services only)

Train ID	Signalling ID	Train Operator	Days Run	Origin	Destination	Arrival Time	Departure Time	Services Run	On Time	% On Time	% within 5m	Time Measured	19/05	20/05	21/05	22/05	23/05	24/05	25/05	26/05	27/05	28/05	29/05	30/05	31
W09395	2N08	SE	Mon-Fri	LONDON CHARING CROSS	GRAVESEND	07:07	07:07	59	46	78%	98.3%	*Departure*	07:07	07:10	07:08	07:07	07:08			N/S	07:07	07:07	07:08	07:07	
W07600	2D09	SE	Mon-Fri	CRAYFORD	LONDON CANNON STREET		07:10	59	52	88.1%	98.3%	*Departure*	07:10	07:10	07:10	07:10	07:10			N/S	07:10	07:10	07:10	07:10	
W07628	2D14	SE	Mon-Fri	GILLINGHAM (KENT)	LONDON CHARING CROSS	07:17	07:17	57	39	68.4%	98.2%	*Departure*	07:17	07:18	07:19	07:17	07:18			N/S	07:17	07:17	07:17	07:17	
W07613	2D11	SE	Mon-Fri	CRAYFORD	LONDON CANNON STREET		07:30	55	50	90.9%	98.2%	*Departure*	07:30	07:31	07:30	07:30	07:30			N/S	07:30	07:30	07:30	07:30	
W09397	2N10	SE	Mon-Fri	LONDON CHARING CROSS	GRAVESEND	07:37	07:37	59	52	88.1%	98.3%	*Departure*	07:37	07:37	07:36	07:37	07:37			N/S	07:41	07:37	07:37	07:37	
W05911	1D50	SE	Mon-Fri	GRAVESEND	LONDON CHARING CROSS	07:38	07:38	57	26	45.6%	98.2%	*Departure*	07:39	07:38	07:41	07:39	07:39			N/S	07:39	CAN	07:39	07:38	
W09079	2M10	SE	Mon-Fri	LONDON CHARING CROSS	CRAYFORD	07:48		59	51	86.4%	96.6%	*Arrival*	07:50	07:51	07:48	07:48	07:48			N/S	07:49	07:49	07:48	07:48	
W07624	2D13	SE	Mon-Fri	CRAYFORD	LONDON CANNON STREET		07:50	58	52	89.7%	100%	*Departure*	07:51	07:52	07:50	07:50	07:50			N/S	07:50	07:50	07:50	07:50	
W09401	2N11	SE	Mon-Fri	LONDON CANNON STREET	SLADE GREEN	07:50	07:50	58	36	62.1%	98.3%	*Departure*	07:50	07:50	07:51	07:51	07:51			N/S	07:52	07:51	07:50	07:51	
W05912	1D52	SE	Mon-Fri	GILLINGHAM (KENT)	LONDON CHARING CROSS	07:56	07:56	59	26	44.1%	91.5%	*Departure*	07:58	07:58	07:59	UNK	07:57			N/S	07:58	07:59	07:58	07:58	
W09402	2N12	SE	Mon-Fri	LONDON CHARING CROSS	GRAVESEND	08:04	08:05	59	48	81.4%	96.6%	*Departure*	08:04	08:06	08:05	08:04	08:05			N/S	08:05	08:07	08:05	08:07	
W09078	2M09	SE	Mon-Fri	LONDON CANNON STREET	CRAYFORD	08:08		54	50	92.6%	98.1%	*Arrival*	08:07	08:06	08:05	08:07	08:06			N/S	CAN	08:08	08:06	08:06	

Figure 7.9. Tabular visualisation for trains running via Crayford with all journeys listed at right, all journeys conducted from 19/5/14 to 24/8/14

APPENDIX 6: TASK-BASED TESTING FEEDBACK

USABILITY TESTING – TASKS REQUESTED FOR TESTERS

***Scenario 1**: User wants to know what time they can expect to arrive at their destination*

Commuter gets on the train at Crayford, travels to London Charing Cross via Sidcup at about 7:15am each day for work. Please report the average expected time of arrival at London Charing Cross, using both the tabular and graph visualisations.

***Scenario 2:** User wants to know with 90% confidence what time their train will arrive.*

Passenger boards a train at Brighton at 9:30 to catch a plane at Luton Airport. The passenger is catching a plane that departs Luton Airport at 12:40. Assuming that all flight passengers must arrive at check-in one hour before departure, do testers feel confident of catching this train in order to arrive at the airport in time?

***Scenario 3**: User wants to know how many trains pass through a station in any given time period*

Railway manager works at London Blackfriars station, wants to know the number of trains that travel through during the morning peak hour (7am – 10am), as well as how many of those trains arrive on time or within five minutes.

HIERARCHICAL TASK ANALYSIS – EXPECTED FLOW OF BEHAVIOUR

TASK: User wants to know what time they can expect to arrive at their destination

0. In order to find out the average time of arrival for their train
    1. Open web browser
    2. Load web page
    3. Click “Select To and From Stations” link
    4. Select their origin station
    5. Select their destination station
    6. Select any intermediate station
    7. Enter the earliest time for train departure
    8. Enter the latest time for train departure
    9. Select mode of visualisation
        i. Select tabular visualization
            1. Find their train on the results
            2. Look for the average time column
        ii. Select graphic visualization
            1. Find their train on the graph
            2. Hover over destination station
                a. Select scheduled path
                b. Select average path
            3. Read average arrival time for tooltip
    10. Close web browser

Plan 0: do 1-2-3-4-5-6-7-8-9-10. Points 4-5-6-7-8 can be done in any order.
Plan 9: do 9.i-9.ii. If no trains found, return to point 3.
Plan 9.i: do 9.i.1-9.i.2.
Plan 9.ii: do 9.ii.1-9.ii.3.
Plan 9.ii.2: do 9.ii.2.a or 9.ii.2.b. Both selections contain the same information.

TASK: User wants to know how often their chosen train is 10 minutes late or more

0. In order to find out the average time of arrival for their train
    1. Open web browser
    2. Load web page
    3. Click “Select To and From Stations” link
    4. Select their origin station
    5. Select their destination station
    6. Select any intermediate station
    7. Enter the earliest time for train departure
    8. Enter the latest time for train departure
    9. Select mode of visualisation
        i. Select tabular visualization
            1. Find their train on the results
            2. Look for the average time column
        ii. Select graphic visualization
            1. Find their train on the graph
            2. Hover over destination station
                a. Select scheduled path
                b. Select average path
            3. Read average arrival time for tooltip
    10. Close web browser

Plan 0: do 1-2-3-4-5-6-7-8-9-10. Points 4-5-6-7-8 can be done in any order.
Plan 9: do 9.i-9.ii. If no trains found, return to point 3.
Plan 9.i: do 9.i.1-9.i.2.
Plan 9.ii: do 9.ii.1-9.ii.3.
Plan 9.ii.2: do 9.ii.2.a or 9.ii.2.b. Both selections contain the same information.

TASK: User wants to know how many trains pass through a station in any given time period

0. In order to find out the number of trains passing through a station
    1. Open web browser
    2. Load web page
    3. Click “Traffic for One Station” link
    4. Select their origin station
    5. Enter the earliest time for train departure
    6. Enter the latest time for train departure
    7. Click “Search Trains”
    8. Count number of trains
    9. Close web browser

Plan 0: do 1-2-3-4-5-6-7-8-9. Points 4-5-6-7 can be done in any order.

USER SURVEYS

USER SURVEY – DATA VISUALISATION

Name of Researcher
Matthew Griffin
Title of study
Data visualization for rail commuters and railway operators

Please read this form fully and carefully, and complete the form to the best of your ability. Do not mark this page with any identifying information. If you do not understand anything and would like more information, do not hesitate to ask for clarification.

	Definitely agree	Somewhat agree	Neither agree nor disagree	Somewhat disagree	Definitely disagree
The table data was easy to understand		✓			
The graph data was easy to understand		✓			
I felt comfortable looking at the table		✓			
I felt comfortable looking at the graph		✓			
The data in the table was useful to me		✓			
The data in the graph was useful to me		✓			
The process of selecting my journey was straightforward	✓				
The statistical information reported was interesting	✓				
I would feel comfortable using this system to choose my journey(s) in the future		✓			

Please add any additional comments about the visualisations in the section below (optional).

- Summary data, eg totals, # rows, etc.
- An ability to select 'Engineer' or 'Public', to show eg calling points only, or stations only, to limit # of things displayed in drop downs.

Thank you for your time and input today.

USER SURVEY – DATA VISUALISATION

Name of Researcher
Matthew Griffin
Title of study
Data visualization for rail commuters and railway operators

Please read this form fully and carefully, and complete the form to the best of your ability. Do not mark this page with any identifying information. If you do not understand anything and would like more information, do not hesitate to ask for clarification.

	Definitely agree	Somewhat agree	Neither agree nor disagree	Somewhat disagree	Definitely disagree
The table data was easy to understand	✓				
The graph data was easy to understand	✓				
I felt comfortable looking at the table	✓				
I felt comfortable looking at the graph	✓				
The data in the table was useful to me	✓				
The data in the graph was useful to me		✓			
The process of selecting my journey was straightforward	✓				
The statistical information reported was interesting	✓				
I would feel comfortable using this system to choose my journey(s) in the future	✓				

Please add any additional comments about the visualisations in the section below (optional).

- Add legend on graph
- historic data should be second layer of info. First layer should be summary..
- clarify how to input time (mmhh)
- Add total # of trains on the page

Thank you for your time and input today.

USER SURVEY – DATA VISUALISATION

Name of Researcher
Matthew Griffin
Title of study
Data visualization for rail commuters and railway operators

Please read this form fully and carefully, and complete the form to the best of your ability. Do not mark this page with any identifying information. If you do not understand anything and would like more information, do not hesitate to ask for clarification.

	Definitely agree	Somewhat agree	Neither agree nor disagree	Somewhat disagree	Definitely disagree
The table data was easy to understand	✓				
The graph data was easy to understand	✓				
I felt comfortable looking at the table	✓				
I felt comfortable looking at the graph	✓				
The data in the table was useful to me	✓				
The data in the graph was useful to me	✓				
The process of selecting my journey was straightforward	✓				
The statistical information reported was interesting	✓				
I would feel comfortable using this system to choose my journey(s) in the future	✓				

Please add any additional comments about the visualisations in the section below (not compulsory).

- It was very accurate about timings.
- I personally prefer using tables, however the graph was straightforward and easy to understand
- It was interesting to have a reason for train cancellations, rather than just the train is cancelled.
- Red and green on the table data was instantly and clearly identifying late or on time.

Thank you for your time and efforts today.

USER SURVEY – DATA VISUALISATION

Name of Researcher
Matthew Griffin
Title of study
Data visualization for rail commuters and railway operators

Please read this form fully and carefully, and complete the form to the best of your ability. Do not mark this page with any identifying information. If you do not understand anything and would like more information, do not hesitate to ask for clarification.

	Definitely agree	Somewhat agree	Neither agree nor disagree	Somewhat disagree	Definitely disagree
The table data was easy to understand	✓				
The graph data was easy to understand		✓			
I felt comfortable looking at the table	✓				
I felt comfortable looking at the graph	✓				
The data in the table was useful to me	✓				
The data in the graph was useful to me		✓			
The process of selecting my journey was straightforward		✓			
The statistical information reported was interesting		✓			
I would feel comfortable using this system to choose my journey(s) in the future		✓			

Please add any additional comments about the visualisations in the section below (not compulsory).

- Legend on Graph missing.

Thank you for your time and efforts today.

USER SURVEY – DATA VISUALISATION

Name of Researcher
Matthew Griffin
Title of study
Data visualization for rail commuters and railway operators

Please read this form fully and carefully, and complete the form to the best of your ability. Do not mark this page with any identifying information. If you do not understand anything and would like more information, do not hesitate to ask for clarification.

	Definitely agree	Somewhat agree	Neither agree nor disagree	Somewhat disagree	Definitely disagree
The table data was easy to understand	✓				
The graph data was easy to understand	✓				
I felt comfortable looking at the table	✓				
I felt comfortable looking at the graph	✓				
The data in the table was useful to me	✓				
The data in the graph was useful to me	✓				
The process of selecting my journey was straightforward	✓				
The statistical information reported was interesting	✓				
I would feel comfortable using this system to choose my journey(s) in the future	✓				

Please add any additional comments about the visualisations in the section below (not compulsory).

- It would be helpful to see a summary of all the train data when looking at the traffic at one station as its a lot of information to digest.
- Graphical analysis was helpful but still yielded the same decision as tabular data but I got to the decision quicker.

Thank you for your time and efforts today.

- platform data would be useful
- direction information useful also
- average arrival time useful
- base each train favour of summary

T3

- straightforward
- time format
- identified data & non-weekend
- summary data identified
- expected "about 8:03am"
- useful for past trains
  need/great to identify lateness
- cancellation info "interesting"
- legend required
- table "more accurate"
- clear about time variance
- station traffic clear to identify late / not late
- more cancellations late in the morning
- capitalise first letter on cancellation

T1

- time format needs to be
- meaningless data
- services run - unclear
- appeal header via site. stop
- define black/blue lines
- line variation - data portrayal
- highlights ~~#hour~~ that skewed avg.
- 11:30 "guessed"
- no total on station page.
- perhaps newest first

T2

- "to/from/via" selection
- time format
- via not working for table
- how many data points represented
- expand axes on screen
- data axes issue
- number of trains

PERSONAL NOTES

TS

- full stop for time
- useless arrival time wrong
- makes trains not run under
- delays at H/Green
- colours not obvious
- useful to know range - graph
- arrive just before 8 am
- scale could be adjustable
- not comfortable with being smelly late
- graph doesn't impact decisions
- identified number of trains
- 80% of trains arrive within 5m
- no overall data
- services gradually worsen over time
- top line summary & aggregation would be useful

- "LONDON" included as main station name
- time format again
- pic train & next train
- past data not essential
- arrive by time selected
- rough range of times.
- blue line not explained
- will tell panel from lines
- needs an "around" button
- risk - averse to missing connection
- graph shows worst - case better
- low relevance for public
- too general, too much data
- customer service benefit
- data is something wrong
- Logo trains are later