Contents

[APPENDIX A: REQUIREMENTS SPECIFICATION 2](#_Toc355349872)

[APPENDIX B: DESIGN SPECIFICATION 12](#_Toc355349873)

[APPENDIX C - TESTING 22](#_Toc355349874)

[APPENDIX D – CODE SUMMARY 55](#_Toc355349875)

[APPENDIX E – MINUTES 71](#_Toc355349876)

[Meeting 1 71](#_Toc355349877)

[Meeting 2 73](#_Toc355349878)

[Meeting 3 75](#_Toc355349879)

[Meeting 4 77](#_Toc355349880)

[Meeting 5 79](#_Toc355349881)

[Meeting 6 81](#_Toc355349882)

# APPENDIX A: REQUIREMENTS SPECIFICATION

Project Name: *Metro Train Timetable*

Team Number 3

Document Information – Requirements Document

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | | |
| Project Name: | Metro Train Timetable | | |
| Prepared By: | Gary Carr, Chris Kerr, Kefu Li, Jasjot Mattu, Antao Xu | Preparation Date:11/12/2012 |  |
| Email / Phone: | [g.carr@newcastle.ac.uk](mailto:g.carr@newcastle.ac.uk) |  |  |
| Document Version No: | 1.3 | Document Version Date: 02/5/2013 |  |
|  |  |  |  |

Version History

| Ver. No. | Ver. Date | Revised By | Description |
| --- | --- | --- | --- |
| 1.0 | 11/12/2012 | Gary Carr | Document compiled and submitted for assessment. |
| 1.1 | 12/4/2013 | Gary Carr | Revision based on feedback from Steve Riddle |
| 1.3 | 02/5/2013 | Chris Kerr | Some grammar changes. |

Terms used in this document are described on page 11.

Analysis process

The team assignment is to develop a system for the Metro where the fastest journey time would be displayed to the user. This in itself can be one of endless assumption, so firstly we had to decide what exactly we wanted to do and how we would approach this. As all members had used rail travel significantly, we were in agreement that we considered ourselves domain experts. Experts in that we all had initial thoughts and ideas of what we wanted from railway software and had all dealt with railway software in some shape or form previously. Therefore following from this we engaged in brainstorming as well as the forming of semantic nets, to give initial ideas as to what the domain would involve and whom the users and stakeholders would be.

From this start we then began to look at other systems and how they operated, specifically those of the National Rail Network[[1]](#footnote-1) and the Eastcoast Mainline[[2]](#footnote-2). These systems were close to what we were initially thinking of how our system would look and act like as these systems allowed for users to plan and find trains for their journeys.

Summary

The client has requested development of a web based system which calculates the most efficient train route dependent on the users starting and destination points, and date and time of travel. It is essential that the website be useable by the majority of the population, and the instructions be clear and easily understood. The design must allow for future changes in the train network as new stations and timetables are implemented.

Environment

The train network we are creating has 10 stations, with a potential in the future to add new stations, routes, or change journey times. There are 4 crossover stations where a traveller may change to another train to complete their journey. The website will always return the fastest route possible, and return the times the train departs, and where a transfer time will occur.

Currently the majority of users will either search a website to find the best route, or ask at the station itself. Websites in this sector are very well developed, for example on [www.thetrainline.co.uk](http://www.thetrainline.co.uk) a journey can be searched with less than 10 keystrokes and clicks.

There are three modifications to the current metro network.

* The number of stations required to calculate journey time has been set to ten.
* A fast line has been implemented between Monument and Whitley Bay.
* A tunnel has been built between North and South Shields.

Stakeholder summary

The end user - The person who is travelling. Every aspect of the design has to be tailored to fit their requirements and expectations so they become repeat users.

* Rival websites – They have become the norm for users over the past 5 years and to attract people to our own website we need to match where possible the simplicity and functionality of these sites.
* Nexus (Metro operators) - Needed to ensure the client has updated lists of timetables and station

Project scope is the process of ensuring that the project deliverables and project boundaries are known to those about to start the project[[3]](#footnote-3). For the purposes of this project the deliverables of the project, essentially that which is understood to be within the scope, are as follows:

In Scope

* The development of a web based system for the client, essentially the purpose of the project
* To ensure the website has the ability to calculate the routes depending on various users’ inputs, such as dates, times and return journeys.
* For simplification the database only includes ten stations on the network. The client can add additional stations to this and a platform to add through MySQL and will not need to modify any of the PHP coding to continue to return the correct results.
* Making sure the project is done within the timeframe to be achieved through meeting milestone targets and regular meetings.
* Additional information of the areas around the stations will be provided.

These are targets which we want to remain within, however it is important to documents what the boundaries of the project are and what is not expected. These can be identified as follows.

Out of scope

* The website is for train routing only. It is not concerned with train booking or train ticket purchasing.
* Following on from this, the software is designed for a website only. The team does not have to construct a hardware based version for ticketing booths or ticket machines on various train platforms. Therefore the team doesn’t have to consider the writing of staff training manuals or factor in potential staff and or ticket machine shortages at various stations.
* There is no requirement to advertise or try to profit from the website
* The software has no interaction with the trains themselves, the software team is given a set of times and routes and updates the web interface accordingly. The team does not have to implement any sort of real time adjustment software and doesn’t correspond with potential delays that could happen and the train times that are show at each station.

Hardware Platforms

The business requirements are very specific regarding the hardware available. The hardware to be used is chosen because that is what is available and the business requires us to use. The website will be hosted on a MySQL database accessible via PHP. The system has two Intel Xeon processors (dual-core).

The hardware to be used will meet our requirements as it meets the business expectations and was submitted to a series of performance tests including CPU, Disk and Memory tests which measure allocating and accessing memory speed and efficiency.

Software Platforms

The software chosen will ensure that the user would not have to download additional plugins for their web browser e.g. Flash and would still be viewable on a smaller mobile device.

The software used to calculate the quickest journey route will be PHP querying a MySQL database. This returns a PHP file to display the final results. Other languages such as JavaScript, JQuery, Ajax and CSS, will be explored to see if it can add better functionality and display to the site, with a goal that the website runs correctly on all the latest versions of the most popular web browsers. These web browsers are listed in the non-functional requirements.

The integrated development environment the team will use will be Eclipse, with plug-ins for the relevant languages and sub-eclipse to upload onto the server. Eclipse is the prime choice as it is a proven industry standard for development, and the vast range of high quality plug in’s will shorten programming time.

Functional Requirements

The functional requirements have been detailed in the table below. For each requirement it has been assessed if it is a high, medium or low priority for inclusion in the first version of the website.

|  |  |  |
| --- | --- | --- |
| F1 | Display minimum transfer time between trains at a specific station | High |
| F2 | Display the fastest route between two stations | High |
| F3 | Display the complete journey time | High |
| F4 | Record all train stations and train times | High |
| F5 | Provide a web based interface to the user | High |
| F6 | Display a return journey | Medium |
| F7 | Display a selection of routes between two stations | Medium |
| F8 | Display direct train routes by station | Medium |
| F9 | Allows customer feedback on system | Low |
| F10 | Display information about station area | Low |
| F11 | User can print instructions for train journey | Low |
| F12 | Users can print train map | Low |
| F13 | System allows for users with visual impairments | Low |

Non-functional requirements

The non-functional requirements have been detailed in the table below. All non-functional requirements are assessed as high priority.

|  |  |
| --- | --- |
| NF1 | The system has complete real-time information |
| NF2 | System displays consistent results |
| NF3 | Results can be obtained in less than 10 key strokes/clicks |
| NF4 | The system is available on the most commonly used browsers (Chrome, IE, Firefox, Safari)[[4]](#footnote-4) |
| NF5 | Website can handle multiple users at any one time |

Considerations

It is expected that the train network we are providing for will operate without unexpected occurrences. A summary of the assumptions we have made of the network are as follows

* All lines are bidirectional
* Trains stop at every station
* Trains always take the same time to run between two given stations
* Trains are never late
* Trains always stop for 2 minutes at every station
* There is an infinite number of tracks available
* The website is designed to provide information only

Where possible the design of the website will allow for future development to change the programming to engineer solutions when these assumptions happen. If it is determined during coding that any of the above can be accommodated for in the first version of the website then it will be included.

Constraints and Dependencies

* The project is expected to be delivered by May 2013. Every effort will be made to meet all the listed functional and non-functional requirements within this timescale though if an unanticipated situation occurs then the short falling will be documented in a final report.
* Wide scale testing and feedback by real users will be difficult. A sample audience will test the completed project on different browsers for feedback on the functionality.
* The hardware will not be owned by ourselves and cannot be adjusted if our needs require it to be. However this is not expected to be an issue.

Website design   
Index page:

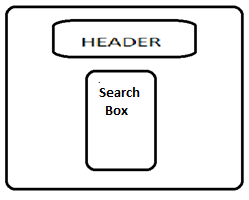


Figure - Mock-up of main page

Header - Will contain navigation bars to Home, Routes, About Us, and Station Information (figure 1).

Search box will contain forms for departure and destination locations and time of travel. The user will input these forms and click search to return a result (figure 2).

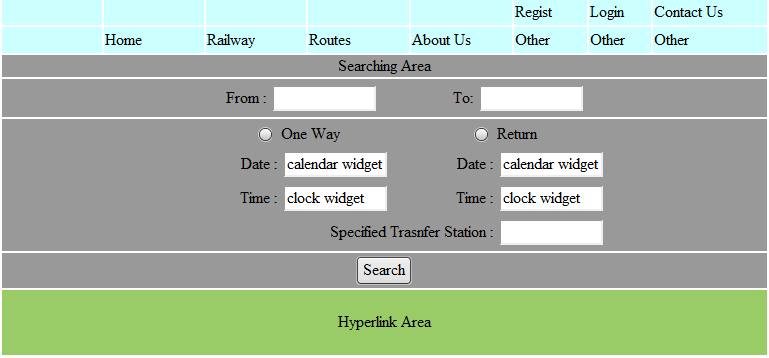


Figure - Inputting search query

Results page:

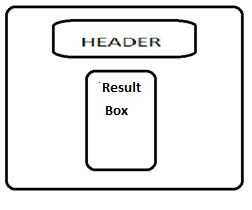


Figure - Mock-up of results page

Header - Will contain navigation bars to Home, Routes, About Us, and Station Information.

Results box – Will output the journey details to the user (figure 4).

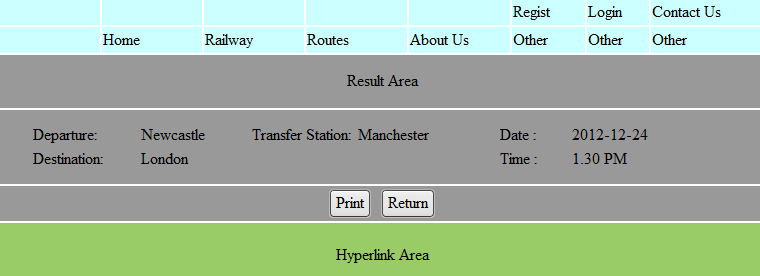


Figure - Results outputted to user

Use Cases

Actions described from high level to low level:

Functional:

1. Display departure and destination

|  |  |
| --- | --- |
| User Action  Choose a departure and a destination     from selection list respectively | System Response  Display departure and destination |

1. Display fastest route and journey time between two stations

|  |  |
| --- | --- |
| User Action  Choose a departure and a destination  Select a depart time Select a return time(Optional) Choose between "Weekday" and "Weekend" Click 'SEARCH FASTEST ROUTE' button | System Response  Check train database to find out fastest route and time Display fastest journey time  Click 'FULL JOURNEY DETAILS' button to see details of fastest route |

1. Display delay time and transfer station

|  |  |
| --- | --- |
| User Action  Choose a departure and a destination  Select a depart time Select a return time(Optional) Choose between "Weekday" and "Weekend" click 'SEARCH FASTEST ROUTE' button | System Response  Check train database to find out fastest route and time Display fastest journey time  Click 'FULL JOURNEY DETAILS' button to see delay and transfer details |

1. Timetable searches for journeys in the future

|  |  |
| --- | --- |
| User Action  Staff input all train stations to database | System Response  Save input to database |

1. Provide a web based interface to the user

|  |  |
| --- | --- |
| User Action  Use web based technique | System Response  Execute the web based interface Display the output |

1. System allows for users with blindness

|  |  |
| --- | --- |
| User Action  Hover over text/drop down boxes (user already having text to speech software) | System Response  Output voice |

1. User can see metro map

|  |  |
| --- | --- |
| User Action  Click "Metro Map" button on the home page | System Response  Display the metro map |

1. User can see information on destinations

|  |  |
| --- | --- |
| User Action  Hover over 'Our Destinations' button    on the home page and click location | System Response  Open new tab to information on that station |

9) User can see photos of team members

|  |  |
| --- | --- |
| User Action  Hover over ‘The Team’ button and click team member | System Response  Displays photo |

10) User can contact team members

|  |  |
| --- | --- |
| User Action  Hover over ‘Contact Us’ button and click team member | System Response  Opens compose message in E-mail client |

11) User can turn to homepage of Newcastle University

|  |  |
| --- | --- |
| User Action  Click 'Newcastle University' button on the    home page | System Response  Turn to homepage of Newcastle University |

12) Display direct train routes by station

|  |  |
| --- | --- |
| User Action  user choose on the screen | System Response  check train database  display required routes |

13) Users can print train map

|  |  |
| --- | --- |
| User Action  user choose on the screen to print map | System Response  check database  print train map via a typewriter |

14) Display a selection of return journeys

|  |  |
| --- | --- |
| User Action  user choose a start station and a destination | System Response  check database  display all possible return journeys |

Definition of Terms

AJAX – A group of technologies to send and receive data from the server asynchronously (in the background) without interfering with the display.

CSS - Cascading Style Sheets. A language used to describe the presentation semantics of a web page or document.

Database – A structured collection of data organized in a way to support accessing this data.

End-user – The person who will be using the site for information and then getting the train

Functional Correctness – Refers to the input-output behaviour of the program e.g. for each input the program produces the correct output.

Graphical User Interface (GUI) – A user interface that allows users to interact with the system using images rather that text commands.

HTML – Hypertext Mark-up Language. The main mark-up language for displaying web pages and other information that can be displayed in a web browser.

IDE –Integrated Development Environment. A software application that provides comprehensive facilities to developers for software development.

JavaScript – A scripting language commonly used as part of the web browser to create enhanced Graphical User Interfaces (GUI).

JQuery – A JavaScript library that simplifies JavaScript programming.

Multi-Platform – Also known as cross platform. The program can be implemented on different computer platforms e.g. operable on Microsoft Windows, Mac OS X and Linux.

MySQL – Open source relational database management system that runs as a server providing access to a number of databases.

Reliability – Ability of the system to perform and maintain its functions in routine circumstances as well as hostile or unexpected circumstances.

Runtime – The period during which a computer program is executing.  
  
Scripting Language – A high level programming language that is interpreted by another program at runtime. Used to add functionality and graphic displays.   
  
Usability – The ease of use of the software application.

# APPENDIX B: DESIGN SPECIFICATION

Project Name: *Metro Train Timetable*

Team Number 3

Document Information – Design Specification

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | | |
| Project Name: | Metro Train Timetable | | |
| Prepared By: | Gary Carr, Chris Kerr, Kefu Li, Jasjot Mattu, Antao Xu | Preparation Date:25/3/2013 |  |
| Email / Phone: | [g.carr@newcastle.ac.uk](mailto:g.carr@newcastle.ac.uk) |  |  |
| Document Version No: | 1.2 | Document Version Date: 02/5/2013 |  |
|  |  |  |  |

Version History

| Ver. No. | Ver. Date | Revised By | Description |
| --- | --- | --- | --- |
| 1.0 | 25/3/2012 | Gary Carr | Document compiled |
| 1.1 | 27/3/2013 | Gary Carr | Revision based on feedback from team meeting. Nielsen’s Heuristic analysis added. Some changes to grammar. |
| 1.2 | 02/5/2013 | Gary Carr | Some grammar changes. |

Overview of the System

User journey

* The site will be coded in HTML5
* The user will input the journey selections into an HTML page, which will post those variables to a PHP page
* The PHP will query a MySQL database to find all possible train routes
* The PHP will assess if a route exists between the stations, at the time required by the user
* The PHP will print details to the user about the journey

HTML

* To ensure consistency each page will be the same in colour, font and layout
* Minimal information will be displayed to the user to satisfy Nielsen’s heuristic of minimalist design. In the results page detailed information of the journey will be automatically hidden from the user (unless the user requests the extra details). The small train network allows this, however this would be revised if the network became larger
* To satisfy Nielsen’s heuristic of error prevention, drop down boxes will be used so that the user does not have to type in the names of each station. This guarantees no input exceptions can occur
* The user will be able to navigate to any part of the site from the any page
* The HTML will be consist with WC3 web standards

MySQL database

* To prevent duplication or inconsistencies the tables will be linked by foreign keys
* A meaningful column name and primary key will be provided to assist any future developer working with the code
* Templates will be produced to allow addition or removal of stations without the user having to write their own code.

Input Rules

* The website will return only the fastest route for the journey, based on the departure station, destination station, and start time
* The website will return an error message if the journey is not found at the specified time, or the stations entered have been the same
* If a user provides an optional return journey time, then that journey will also be displayed

High level overview of how the functionality and responsibilities of the system are partitioned and assigned to components

Website - Index

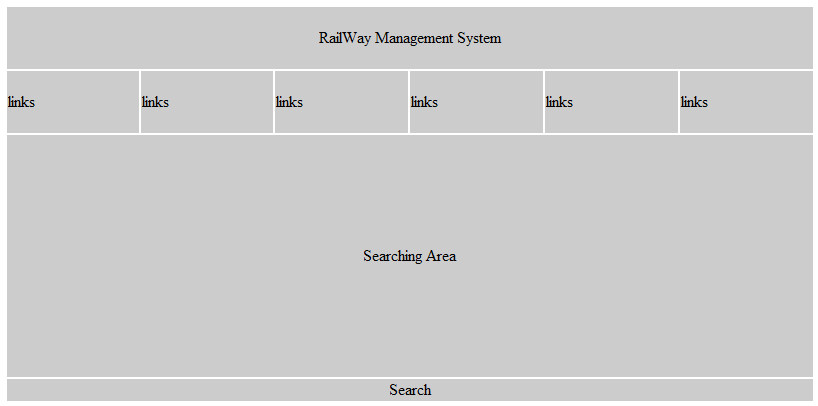


Figure 1 – Index page

In the index page, there are two aspects, the navigation bar and search area. Links make up the navigation bar, such as Metro Search, Destination’s, The Team, Metro Map, Contact us and Newcastle University. Users will be able to click these links to redirect to where they need. Keeping search area clean and simple, so to conform to Nielsen’s heuristics. Users will be able to choose their departure, destination and leave time, and optionally return time, then click search button, and website will redirect to result page, and show information of journey(s).

Website - Result Page

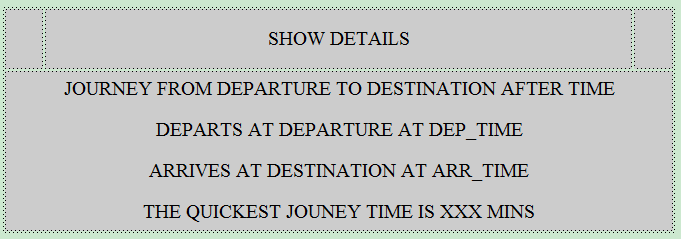


Figure 2 – Results page

The result page shows the summary acquiescently, and hides the details of journey, if users want to check details, they could click "Full Journey Details" button and the website will show all information.

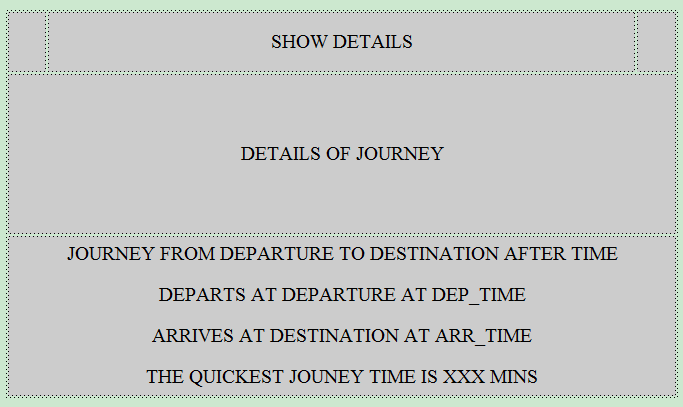


Figure 3 – Result details displayed to users

Database:

For the database part, we have to record details of each station and route, in order to work out the fastest way.

Route Details:

In this table, the database records the full details of each route, for example, the number of route or called the id of path, the departure station, the destination station, certainly, and the cost.

Station Details:

In this table, it records the name of each station. We could record timetable of each route in this table, but we prefer to record details of timing in a new table, because we want to divide timing into working-day and weekend.

Departure Time (working day and weekend):

These two tables record departure time of each route. Because we will create a table record the details of routes, so, we need to connect these two tables by same key word, such as routeNo or pathID. Of course, the departure time is different between working-day and weekend.

All these data structure work using Dijkstra’s algorithm and return the fastest route..



Figure 4 – Structure of the site

**Class Diagram-Metro Search**

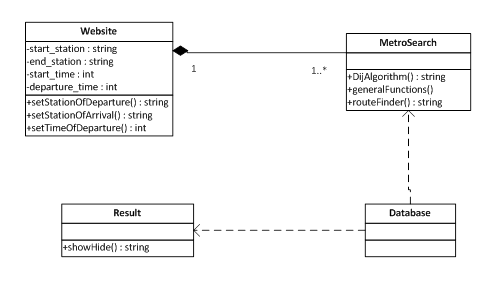


Figure – Class diagrams

The above class diagram shows the actions of the metro search system. System invokes those methods (Website is always the same and users can do search for one or more times as they want) and connects with database to get the value of parameters so that fastest route search can be done. After that, system outputs result to the result page.

Functional Requirements

The functional requirements have been detailed in the table below. For each requirement it has been assessed if it is a high, medium or low priority for inclusion in the first version of the website.

|  |  |  |
| --- | --- | --- |
| F1 | Display minimum transfer time between trains at a specific station | High |
| F2 | Display the fastest route between two stations | High |
| F3 | Display the complete journey time | High |
| F4 | Record all train stations and train times | High |
| F5 | Provide a web based interface to the user | High |
| F6 | Display a return journey | Medium |
| F7 | Display a selection of routes between two stations | Medium |
| F8 | Display direct train routes by station | Medium |
| F9 | Allows customer feedback on system | Low |
| F10 | Display information about station area | Low |
| F11 | User can print instructions for train journey | Low |
| F12 | Users can print train map | Low |
| F13 | System allows for users with visual impairments | Low |

Non-functional requirements

The non-functional requirements have been detailed in the table below. All non-functional requirements are assessed as high priority.

|  |  |
| --- | --- |
| NF1 | The system has complete real-time information |
| NF2 | System displays consistent results |
| NF3 | Results can be obtained in less than 10 key strokes/clicks |
| NF4 | The system is available on the most commonly used browsers (Chrome, IE, Firefox, Safari)[[5]](#footnote-5) |
| NF5 | Website can handle multiple users at any one time |

Dynamic Behaviour of the system

There is a lot of overlap between the functional requirements of the system, therefore in terms of activity diagrams and sequence diagrams, the main functionality of the system will be looked at. This is of course the fastest route between two stations.

Activity Diagrams – F2 Display the fastest route between two stations

[User chooses return time]

Choose Return Time

Choose Departure times

Choose Stations

Choose Day

Search Database

Results Page

[User doesn’t choose return time]

[User does another search]

[User selects full journey]

Full Journey Details

The above activity diagram shows the actions of the user on the system. Firstly the user will be able to choose the stations and the departure times. As the return journey is optional the user does not have to select this but if they do then they select the day (which is only weekend and weekday), which is what they would select if they decided to skip entering a return journey.

With this input, the system searches the database for the respective departure times. These times go through the algorithm and the arrival times and shortest paths are displayed on the results page. From the results page the user will be able to choose to search again, choose full journey details or simply finish their use of the system.

Sequence Diagram – F2 Display the fastest route between two stations

The diagram below shows the sequence diagram of a search for the fastest route between stations. The website will be constantly active throughout system interaction. Once user has inputted the parameters of their search the database becomes active. This searches for the times and the costs between each station. This will then send the information to the algorithm which becomes active and calculates the fastest route based on this database information. This is then sent back to the website, which will display the result of the journey.

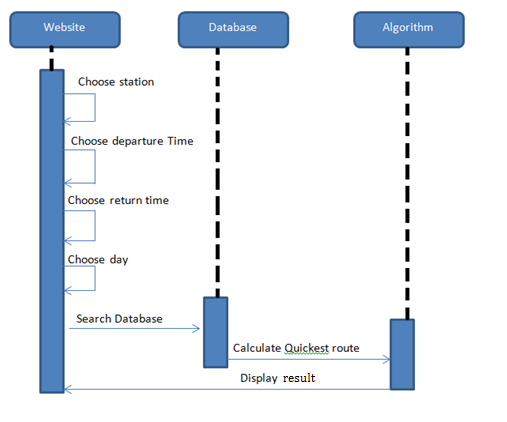


Figure - Calculating fastest route

GUI/Human Interface view

The main goal of the UI was to make the user’s interaction as easy and efficient as possible. The website will guide the user subtly through the process and have minimal amounts of distracting information, which could detract from the process or confuse the user.

When designing the website several questions were considered –

* What are the user’s goals
* What is the user’s skill and experience
* What is the desired result of entering information

The GUI will assist the user to achieve their goals from using the website. No distracting information or images will hide the desired pathway the user has to take to get a train time. The GUI will borrow familiar layout from similar existing websites, this makes the website ‘commonplace’ and develops a feeling of understanding and comfort from a recognised format.

The UI will not allow users to enter any information by keyboard; only pointing devices will be allowed to choose from existing drop down menus. This will take the error rate down to very nearly non-existent and will assist with the speed of achieving the user’s goals. The drop down menus will assist any inexperienced users to enter the correct information and will assist experienced users, who can use a pointing device with speed, due to the size and positioning of the buttons.

The minimalist design will maintain the sleek and rapid user view; the design will remain relevant with the colours easily updateable as desired. The colours and layouts used in the initial website are specifically considered for users with viewing difficulties.

Contrasting colours will be used to draw the eye line to the correct drop box menu and enclosing colour boxes placed around drop down menus that apply to each other to demonstrate what needs to be completed. The format of entering information will follow a typical western writing layout, from left to right and top to bottom to complete the form. The colours chosen for the website will be entered into several colour-blindness tests to allow users with this disability to still distinguish between the menus and background easily and to ensure as many users as possible will be able to complete the form.

The website will have a navigation bar underneath the title with functional links. In an left to right format the leftmost link will lead to the search train time form, as this is the most important and most accessed page it needs to be first to be seen. Therefore the user doesn’t need to look through several non-functional links before this form, this increases efficiency and ease of use.

Comparison against Nielsen’s Heuristics

Throughout the design we have attempted to work to the 10 Usability Heuristics for User Interface Design outlined by Jakob Neilson (Usability Inspection Methods, 1994). The principles and the application are listed below.

Visibility of system status – The website will immediately return journey details. In the event of an input error the user will receive a clear concise error message explaining the mistake, and informing of the mistake.

Match between the system and real world – The text used in the site will be appropriate to the environment users expect when searching train journeys. The input labels will be ‘departure’ and ‘destination’, consistent with signs at train stations, and the link buttons will have the minimum appropriate text to explain the function. During testing the jargon used will be evaluated.

User control and freedom - All pages will be accessible from any other page with a single click. If a user wants to click on an external site, it will be opened in a new tab so the user does not lose the current page.

Consistency and Standards – By design the text and functions will be minimum and non-duplicating. The only inconsistency will be the outlay of the results page which will differ if searching a single or return journey. For a single journey the results will be centralised in one box, for a return journey they will be symmetrical in two identical boxes. Both boxes will be clearly labelled to not cause confusion.

Error prevention – All links and scenarios will be thoroughly tested to remove the probability of encountering an error. The website will be designed so that a user cannot input the same departure and destination stations. In areas where it might seem like the user has made a mistake (e.g. if they attempt to search a journey later than the trains are running) a clear message will be outputted explaining why no result has been displayed.

Recognition rather than recall – The website will contain links in buttons, clearly labelled, and separated which stand out against the background. Where a user has to input information, these areas will be clearly marked. The success of this will be determined in testing, and the site amended as necessary.

Flexibility and efficiency of use – Every possible search should be achievable with 10 clicks/keyboard strokes by the user.

Aesthetic and minimalist design – User options will be kept to a minimum when searching a journey, with the return journey marked as an optional extra. When navigating through our header menu’s the full list of options will be hidden within a drop down bar. Text will be tested to ensure only the minimum amount of relevant information is being displayed to the user.

Help user recognize, diagnose, and recover from errors – As with the error prevention, the site will be thoroughly tested to reduce the possibility of errors and in areas where there could be confusion, clear messages will be displayed.

Help and documentation – By design, error testing, and clear output messages this should not be necessary, but this will be re-evaluated after testing.

# APPENDIX C - TESTING

Project Name: *Metro Train Timetable*

Team Number 3

Document Information – Testing

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | | |
| Project Name: | Metro Train Timetable | | |
| Prepared By: | Gary Carr, Chris Kerr, Kefu Li, Jasjot Mattu, Antao Xu | Preparation Date:25/4/2012 |  |
| Email / Phone: | [g.carr@newcastle.ac.uk](mailto:g.carr@newcastle.ac.uk) |  |  |
| Document Version No: | 1.1 | Document Version Date: 02/5/2013 |  |
|  |  |  |  |

Version History

| Ver. No. | Ver. Date | Revised By | Description |
| --- | --- | --- | --- |
| 1.0 | 26/4/2013 | Jasjot Mattu | Document compiled |
| 1.1 | 1/5/2013 | Gary Carr | Addition of external user testing |

The initial testing for the system will be done internally but from a user point of view. The table below shows the requirements and whether the system passed or failed in delivering them. This is followed by a more in depth look at the requirements (complete with test cases) with screenshots.

**Requirements Table**

Functional Requirements – Pass or Fail (give these a number), write expected output and actual output.

|  |  |  |
| --- | --- | --- |
| F1 | Display minimum transfer time between trains at a specific station | Pass |
| F2 | Display the fastest route between two stations | Pass |
| F3 | Display the complete journey time | Pass |
| F4 | Record all train stations and train times | Pass |
| F5 | Provide a web based interface to the user | Pass |
| F6 | Display a selection of routes between two stations | Pass |
| F7 | Display direct train routes by station | Pass |
| F8 | Display a return journey | Pass |
| F9 | User can print instructions for train journey | Pass |
| F10 | Allows customer feedback on system | Pass |
| F11 | Display information about station area | Pass |
| F12 | Users can print train map | Pass |
| F13 | System provides access for users with visual impairments | Pass |

Non- Functional Requirements – Pass or Fail

|  |  |  |
| --- | --- | --- |
| NF1 | System Displays Consistent results | Pass |
| NF2 | Results can be obtained in less than 10 key strokes/clicks | Pass |
| NF3 | The system is available on most commonly used browsers | Pass |
| NF4 | System displays forgiveness | Pass |
| NF5 | Website can handle multiple users at any one time | Pass |

**In-depth look – Internal Testing: Functional Requirements**

**F1: Provide a web based interface to the user**

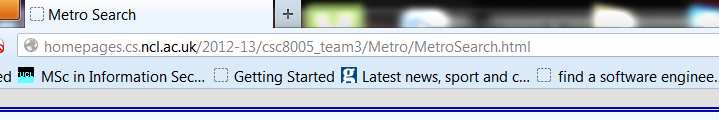
**Test case: Entering URL into Mozilla Firefox browser**

**Expected Output: The website is able to be loaded in all browsers**

**Actual Output:**

Figure 1 shows a screenshot of the website URL being entered into the ‘Firefox’ address bar. Entering this address allows the user to reach the main ‘Metro Search’ website, which is show in figure 2. Therefore as this leads to a web based interface, this functional requirement has been satisfied.

**Fig 1. Showing website address being inputted into address bar**

****

**F2: Display minimum transfer time between trains at a specific station**

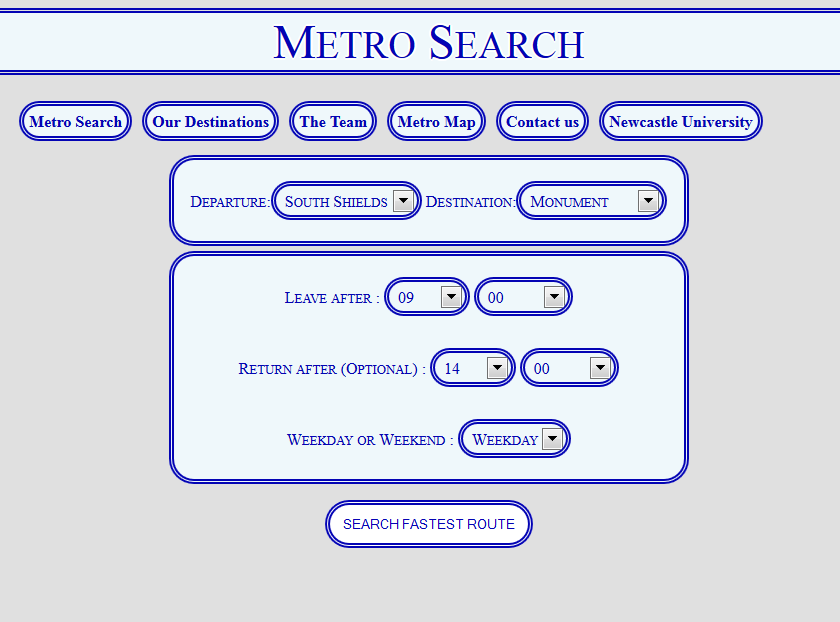
**Test Case: Weekday Departure from South Shields to Monument at 09:00am**

**Expected Output: The minimum transfer time is displayed with no problems**

**Actual Output:**

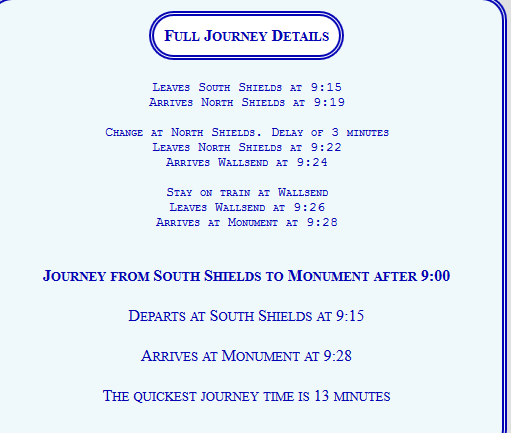
Fig 2 shows a screenshot of the user choosing the departure station as ‘South Shields’ and the destination station as ‘Monument’. Then leaving after 09:00am and returning after 14:00pm on a weekday.

**Fig 2. – Shows the aforementioned journey being inputted into the website**



When they then select ‘Search Fastest route’ it leads to the results page. When the full journey details button is pressed we can see a description of the journey, complete with transfers.

**Fig 3 – The full journey time showing any transfer times**



In this specific example (shown in figure 3), the user would have to change at North Shields, causing a delay of 3 minutes as they change train. So the minimum transfer time in this example is ‘3 minutes’. Therefore with this as evidence, we can say that the system has passed this functional requirement.

**F3: Display the fastest route between two stations**

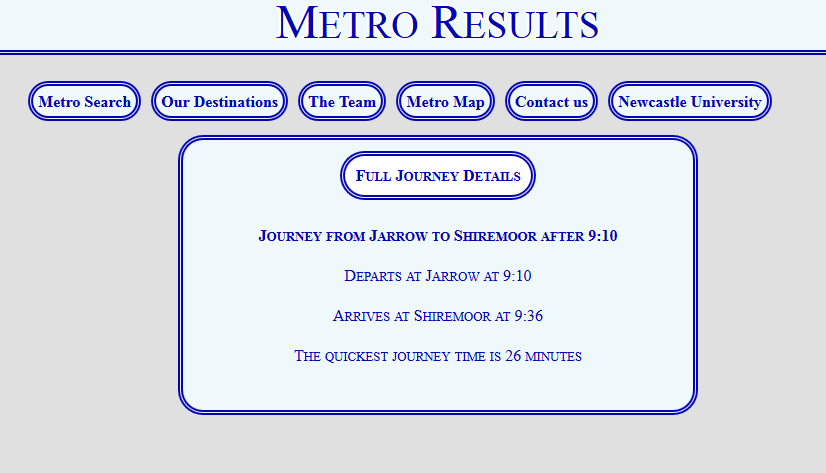
**Test Case: Weekday Journey from Jarrow to Shiremoor at 09:10AM and return at 12:00PM**

**Expected Output: Fastest route is displayed on the webpage**

**Actual Output:**

Figure 4 shows the output of a search from Jarrow to Shiremoor station, here we can clearly see that the quickest journey time between the stations has been found. Therefore this functional requirement has been satisfied.

**Fig 4. – Showing the results of the test case search**



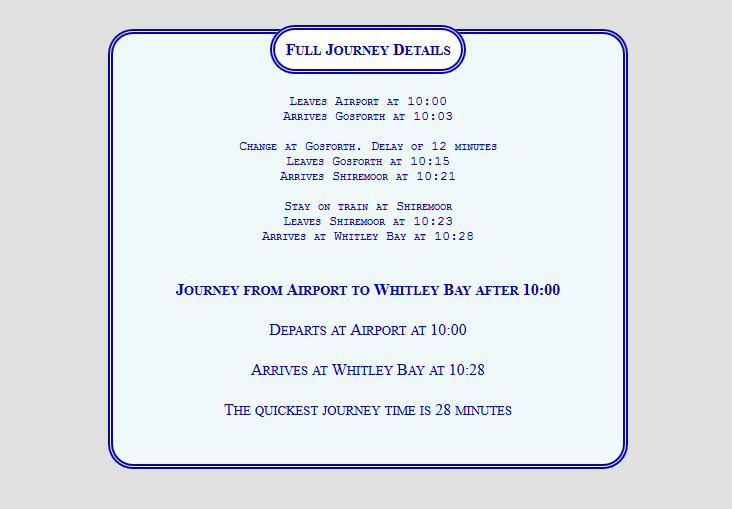
**F4: Display the complete journey time**

**Test case: Weekday Journey from Airport to Whitley Bay at 10:00am**

**Expected Output: Complete journey time is displayed correctly**

Figure 6 below shows the output of the searching for a journey from Airport to Whitley Bay at 10:00am. Once the user selects the ‘Full Journey Details’ they are given a complete breakdown of the journey time, including any transfers and what stations they go through. Therefore this evidences the system displaying the complete journey time and therefore satisfying this requirement.

**Fig 6 – Showing complete journey time**



**F5: Record all train stations and train times**

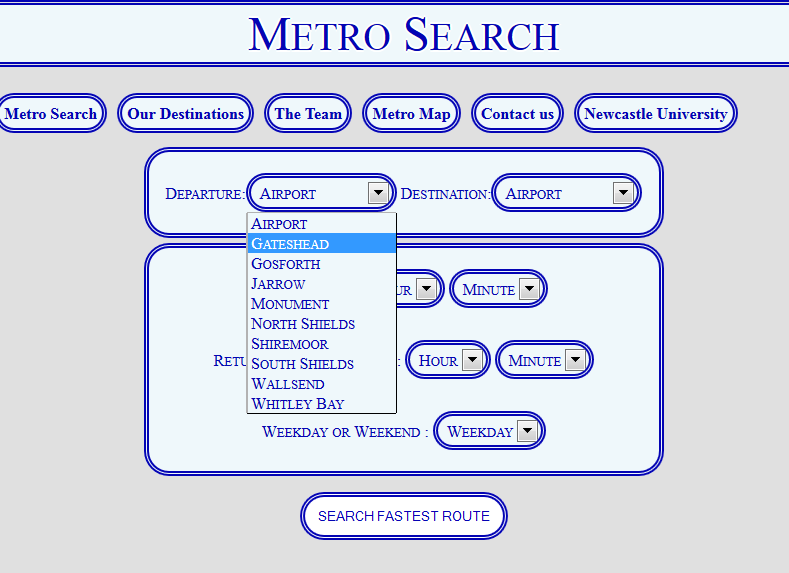
**Test Case: Highlighting stations on Departure drop down box**

**Expected Output: Evidence that train times have been recorded**

**Actual Output:**

On the metro search homepage, once a user selects the drop down box, a list of stations appears (shown in figure 7). This is the same for destination. This effectively serves as a record for all stations on our map. The train times has been evidenced in other test cases previous to this one and in future ones, with various searches already shown to be outputting train times. Therefore, based on this evidence, this functional requirement has been met.

**Fig 7- List of train stations**

****

**F6: Display direct train routes by station**

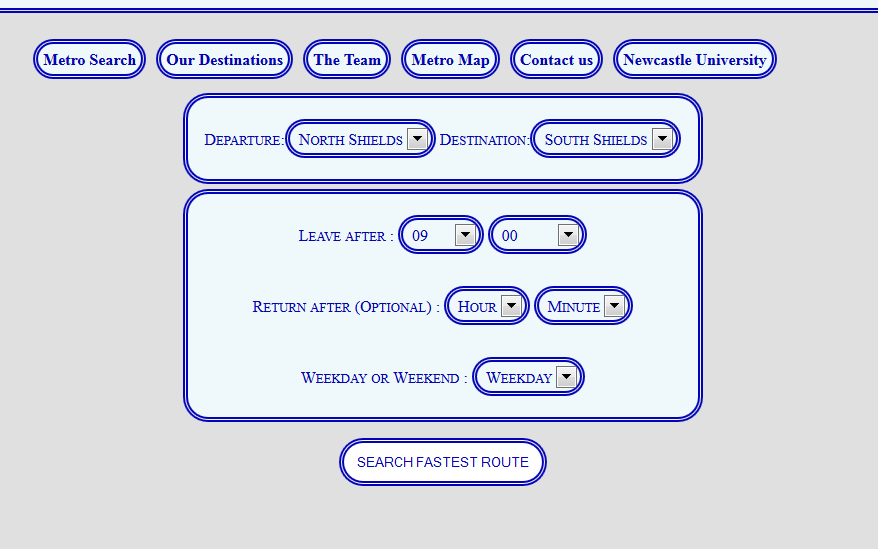
**Test case: Weekday Journey from North Shields and South Shields at 09:00am**

**Expected Output: A direct route is shown with no intermediary stations**

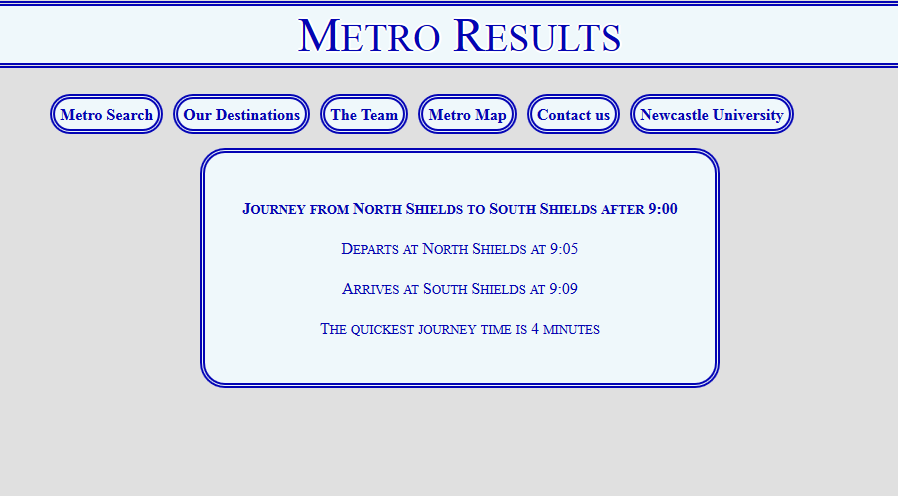
**Actual Output:**

Figure 8 shows a journey being inputted on a weekday at 09:00am from North Shields to South Shields. There is no return journey and the journey takes place on a weekday. Once the search is commenced we see (from figure 9) that it shows the direct route between the two stations and that there are no intermediary stations (therefore not requiring the user to press the full journey button. Therefore this evidences a direct train route, so this requirement has been satisfied.

**Fig 8 – Showing test case departure**

****

**Fig 9 – Showing direct route result**

****

**F7: Display a return journey**

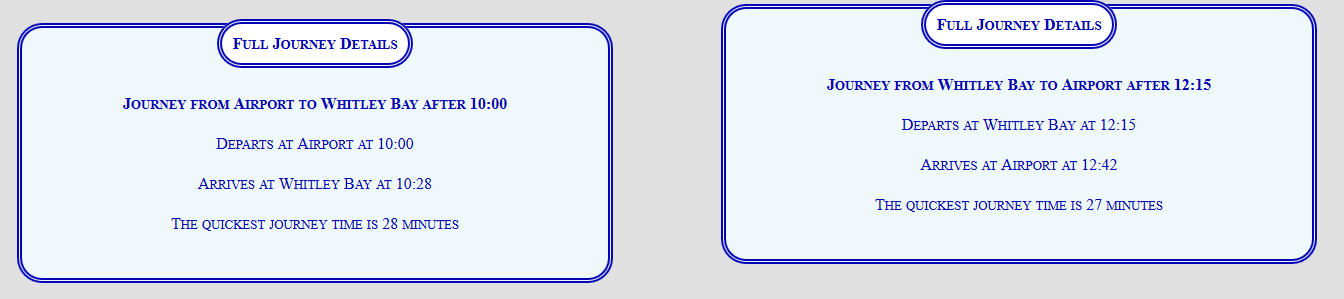
**Test Case: Weekday Journey from Airport to Whitley Bay at 10:00AM and return at 12:15PM**

**Expected Output: Return Journey is displayed**

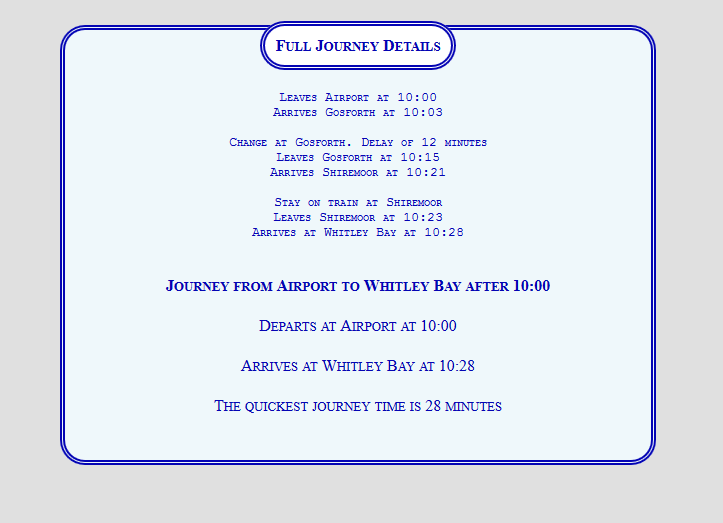
**Actual Output:**

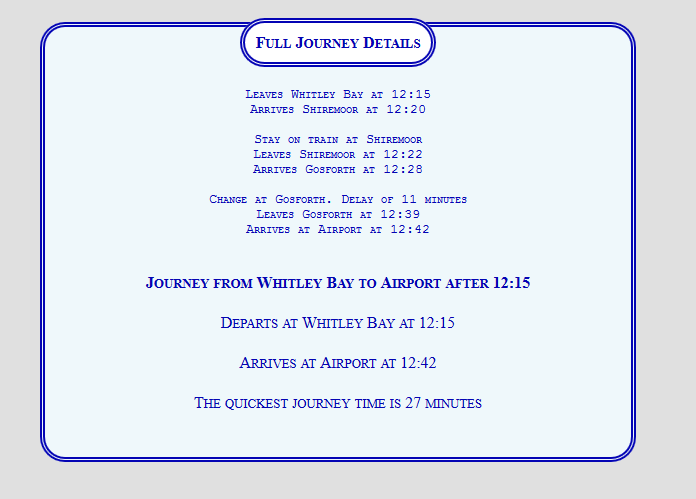
Figure 10 shows a journey from Airport to Whitley Bay at the time of 10:00am and a return at 12:15. Once the ‘Full Journey Details’ button has been pressed then the full information about the return journey is shown, in this case from Whitely to Airport, returning at 12:15pm (shown in figure 11 on the next page). This evidences this requirement and therefore the system has successfully fulfilled it.

**Fig 10 – Showing the return journey from Whitley Bay to Airport**

****

**Figure 11 – Showing the full journey details of these journeys**





**F8: System Allows Customer feedback**

**Test Case: User wants to contacts designers to make a complaint**

**Expected Output: Users can email designers from website directly**

**Actual Output:**

Figure 12 below shows the contact us bar button. Once this has been hovered over a drop list of team member names is opened up, selecting any one of these opens up link to that respective team members contact address. This allows the users to give any feedback they have on the system directly. Therefore this evidences the system meeting this requirement

**Fig 12 – Showing ‘contact us’ bar on website**

****

**F9: Display information about station area**

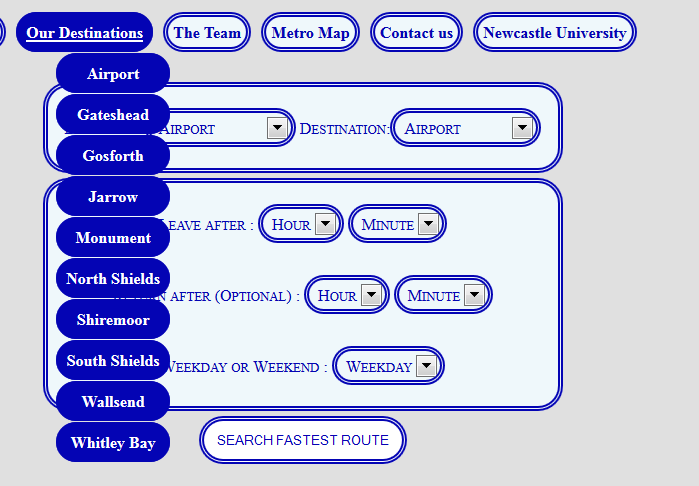
**Test case: User wants to find out information about Gosforth.**

**Expected Output: When clicking on a destination in ‘our destination’ hyperlink opens up and user is taken to external website**

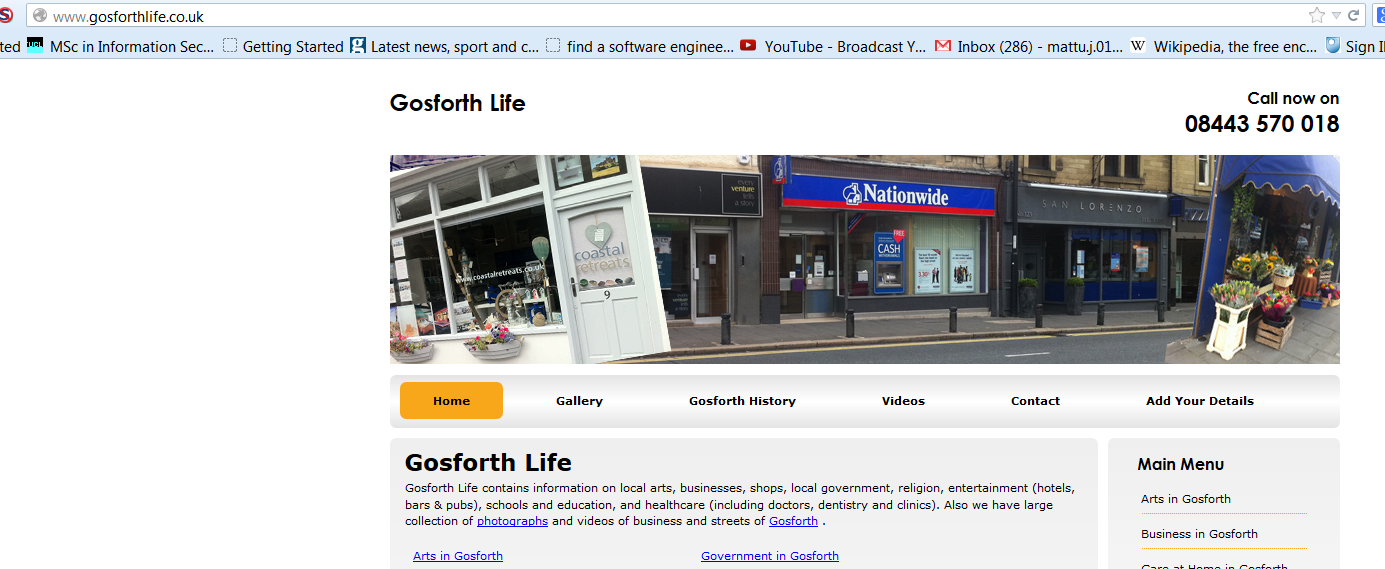
**Actual Output:**

If the user wants to find out about Gosforth, they highlight the cursor over the ‘Our destinations’ tab Shown in figure 13. Selecting Gosforth opens up an external link to an external website that has detailed information about the area. This is the same for all the station. An example of the Gosforth website is shown in figure 14. Therefore with this evidence it can be said that this requirement has been met.

**Fig 13- Showing the our destinations bar**

****

**Fig 14 – Showing an external website with information about Gosforth**



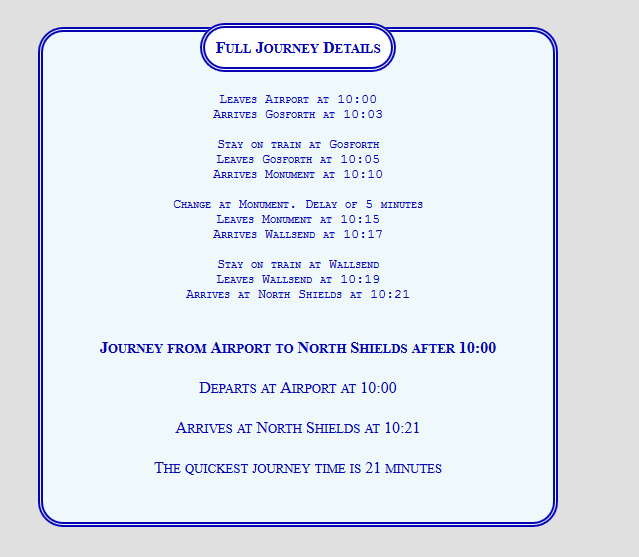
**F10: User can print instructions for train journey**

**Test Case: Users wants to print details of Journey from Airport to North Shields.**

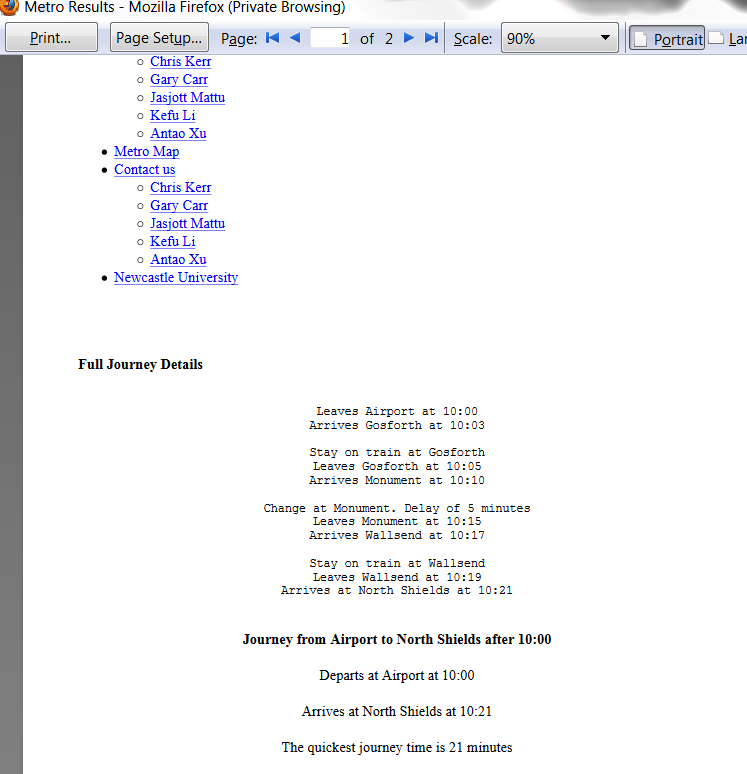
**Expected Output: Users can print page**

**Actual Output:**

Figure 15 shows the results for this journey. The results are presented in this way so that the user can simply just press the print page button, to get the results in printable format, show in figure 16.

**Fig 15 – Full journey time details to be printed**  
  


**Fig 16 – The print page viewpoint, showing the full journey details**



**F11: User can print train map**

**Test Case: Users wants to print map of stations.**

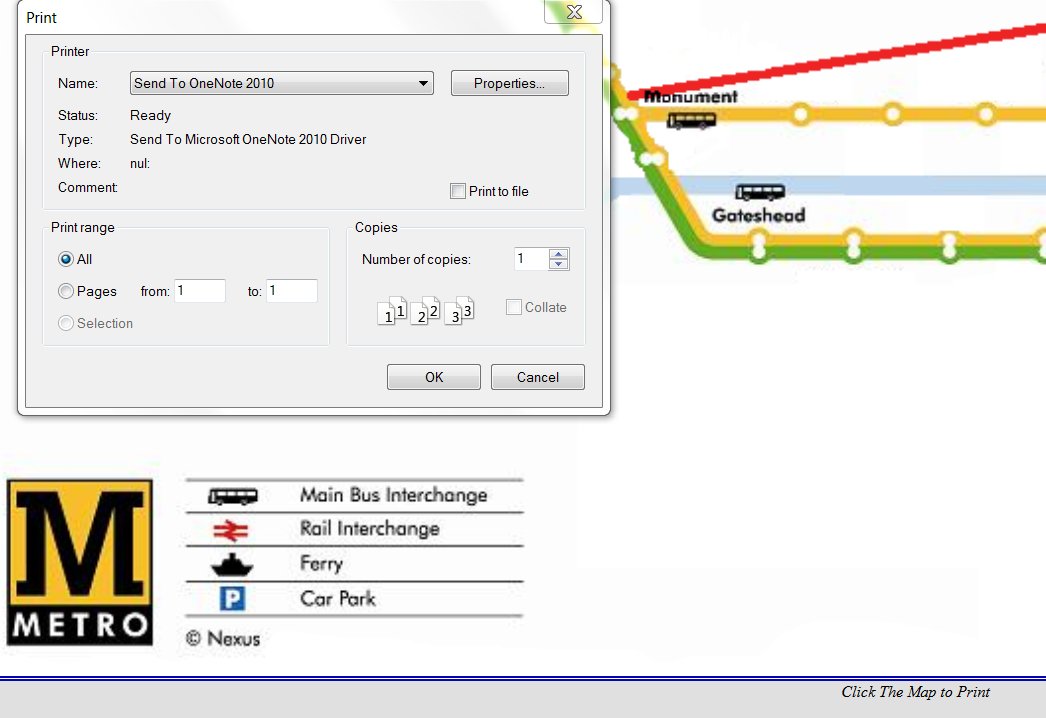
**Expected Output: User can print map**

**Actual Output:**

If the user wants to print the train map, they first click on the ‘metro map’ button located on the website homepage. This takes them to a picture of the train map. Here if they click on the map (as notified by a ‘Click on Map print’) a print -image window pops up. Selecting this prints the map. The evidence for this is shown in figure 17. Therefore with this evidence it can be said that this functional requirement has been met.

**Fig 17-Showing**



****

**F12: System provides access for users with visual impairments**

**Special note: Limitations of visual accessibility testing**

Each feature of our website was tested with freely available software to ensure it was accessible to those with visual impairment. Unfortunately due to time constraints we were unable to do qualitative or quantitative testing with those users and therefore cannot verify that in all circumstances our website meets the functional requirement. If the project were to be developed further we would undertake this research.

**Test Case: User with Colour blindness uses the system**

**Expected Output: The website can be visible when using colour blind filters**

**Actual Output:**

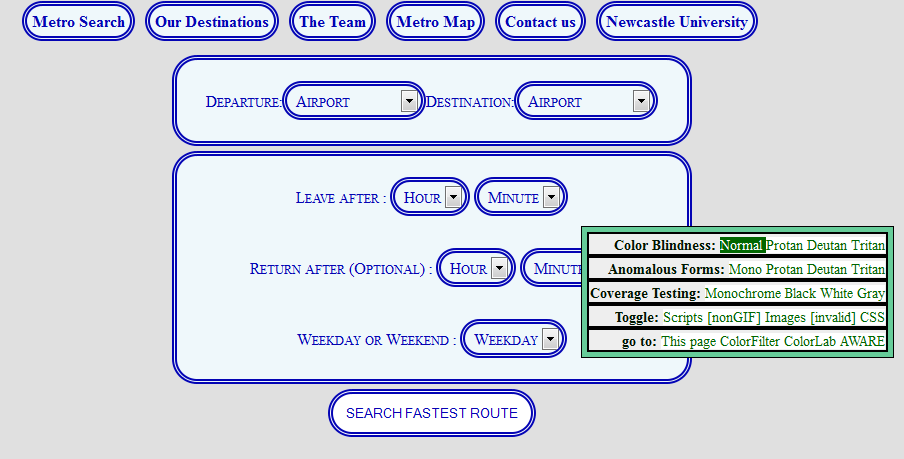
To test the design of the site we used the website <http://colorlab.wickline.org/> which displays the site as people with differing colour blindness would view it. The following conditions were set to satisfy its compatibility (for all types of colour blindness)

* The text must be clearly visible
* The sections must stand out and not blend into the background
* It must be clear where to click for the drop down boxes

Figure 18 is the actual website. Protan and Tritan colour blindness (figure 19 and 21) result in an almost identical view of the site whilst Deutan (figure 20) has a different colour scheme.

The neutral colours chosen in the design has resulted in the text, sections, and drop down boxes being clearly visible in all scenarios and satisfying the requirement of being usable by colour blind users.

**Fig 18- Viewed without colour blindness**



**Fig 19 Viewed with Protan colour blindness**

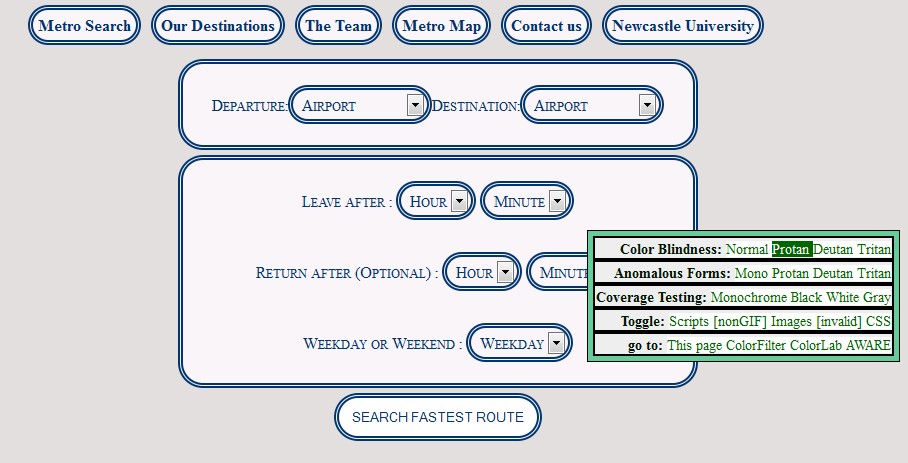


Fig 20 Viewed with Deutan colour blindness

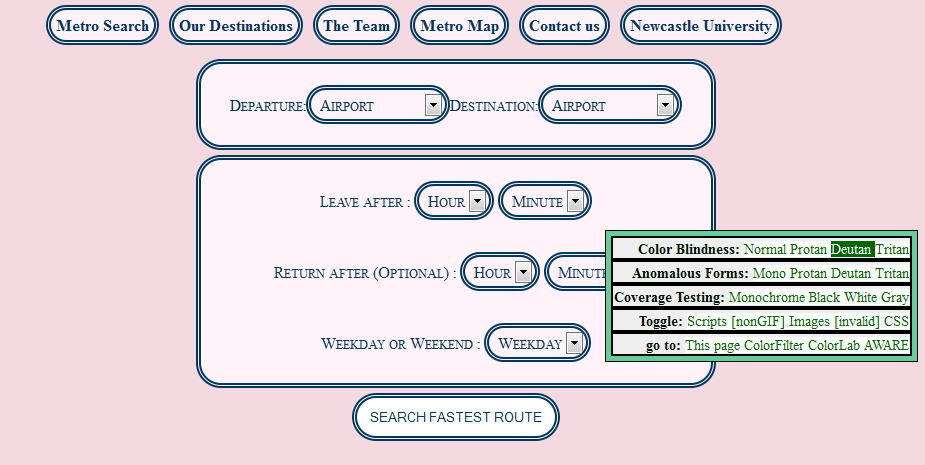
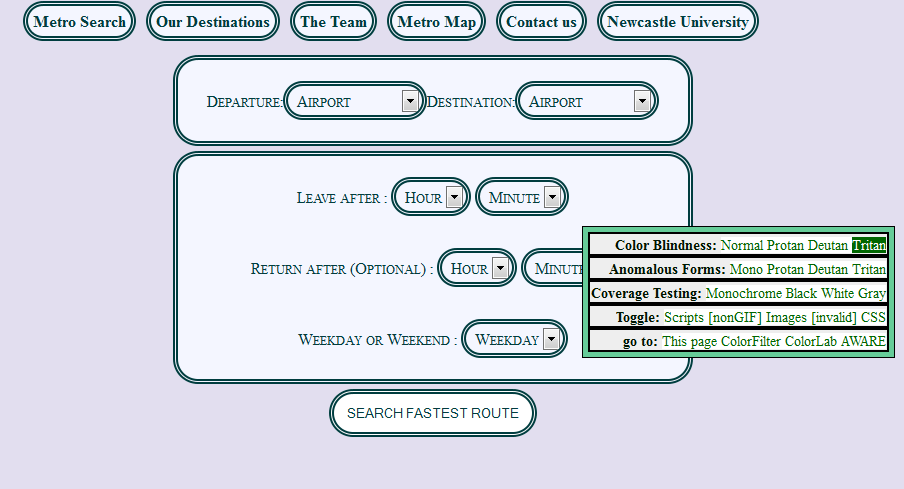


Fig 21 Viewed with Tritan colour blindness



**F13 (Continued) Test Case: Users requires Text to speech**

**Expected Output: Details on the website are read out using computer voice**

**Actual Output:**

Partially sighted users use text to speech editors to announce the wording on the text as they hover over the page. The aim of the website was always to be as simple as possible for any user, and this has had the added benefit that there are very few fields and boxes to be hovered over, reducing irrelevant information announced.

Text to speech was tested using extensions for Firefox and Chrome. Each of the text fields and drop down boxes are spoken correctly, and when tabbed through are announced in a logical order for the user. Therefore this evidences that the system can yet again cater to users with visual impairments.

**Internal Testing - Non-Functional Requirements**

**NF1: System Displays Consistent results**

**Test Cases:**

* **Weekend journey from Airport to Whitely Bay at 09:00am with return at 12:00pm,**
* **Weekend journey from Shiremoor to Monument at 11:30am with return at 13:00pm,**
* **Weekday journey from Jarrow to North Shields at 18:00pm with return at 21:00pm**
* **Weekday journey from Gosforth to South Shields at 20:00pm with return at 22:00pm**

**Expected Output: All results show similar results with no problems**

**Actual Output:**

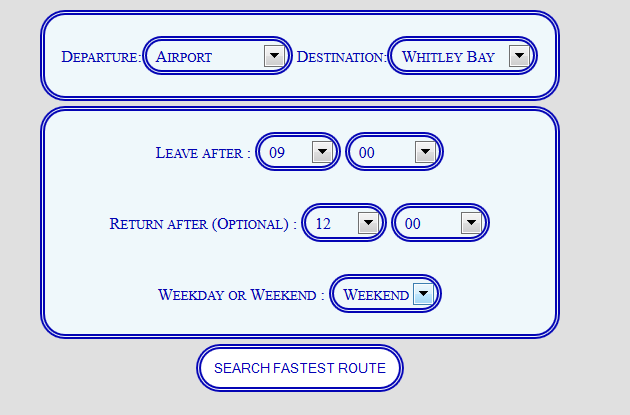
To display consistent results, four test cases are used, two for weekend journeys and two for weekday journeys. Figure 22, shows the first test case, a weekend journey from Airport to Whitely with a time of 09:00pm and a return of 12:00pm. Figure 23 and 24 show the departing and return journeys for these two stations. The second test case is shown in figure 25, a weekend journey from Shiremoor to Monument departing at 11:30 am and returning at 13:00pm. The results of this search are shown in figures 26 and 27. Therefore although the journey times are different, the system is consistently putting out results, satisfying the consistent result requirement for weekend journeys.

The third test case showcases a weekday journey from Jarrow to North Shields at 18:00pm with a return at 21:00pm. This is shown in figure 28. The results of this search are shown in figures 29 and 30, showing the results, albeit in non- expanded form. The fourth test case is a weekday journey from Gosforth to South Shields at 20:00pm with a return at 22:00pm.

This is shown to be inputted in figure 31, with the results in non-expanded form shown in figures 32 and 33. These are consistent with figures 28 to 30 and therefore evidence the consistency of the system.

Therefore from this evidence, it can be said that the system does meet this requirement.

**Fig 22 - Selecting journey from Airport to Whitely Bay at 09:00am and return at 12:00pm**



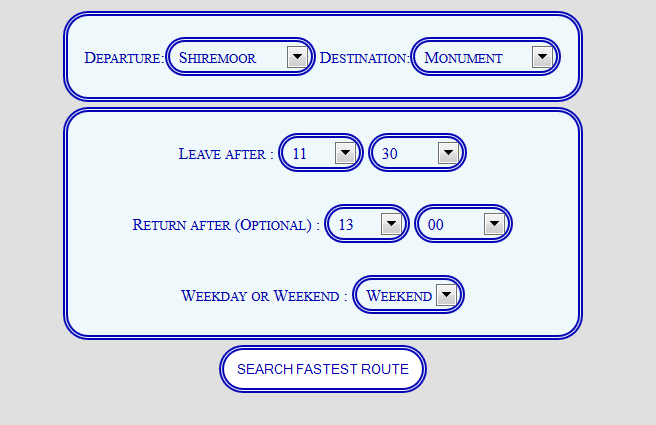
**Fig 23- Departure Journey result from Airport to Whitley Bay at 09:00am**



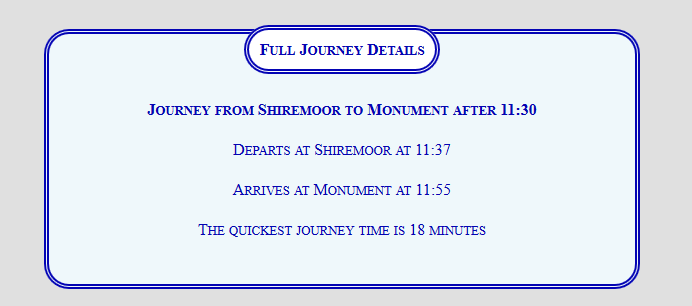
**Fig 24 - Return Journey result from Whitley Bay to Airport at 12:00pm**



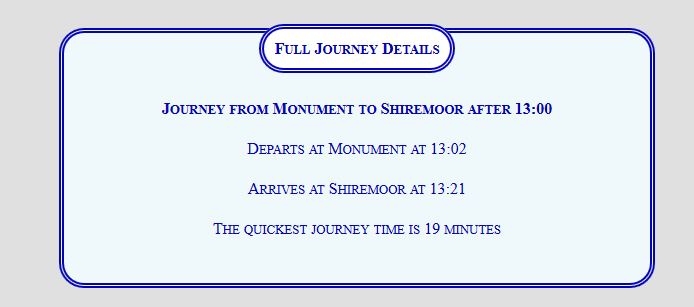
**Fig 25 - Selecting journey from Shiremoor to Monument at 11:30AM and return at 13:00PM**

****

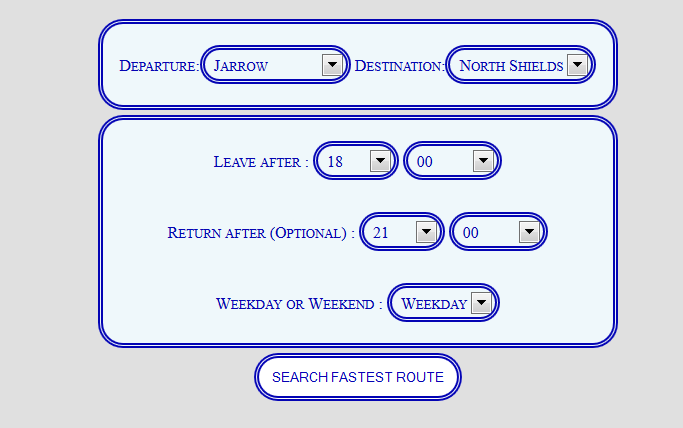
**Fig 26- Departure Journey result from Shiremoor to Monument at 11:30am**

****

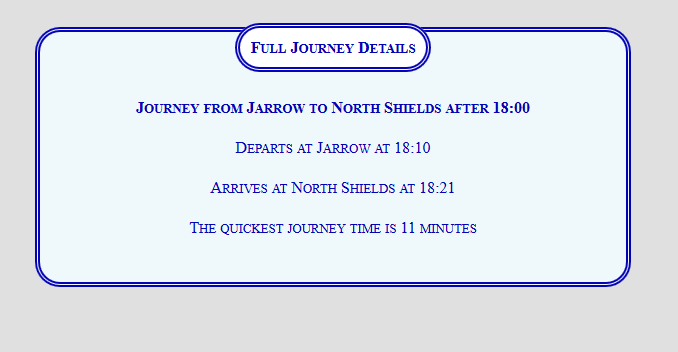
**Fig 27 - Return Journey result from Monument to Shiremoor at 13:00pm**

****

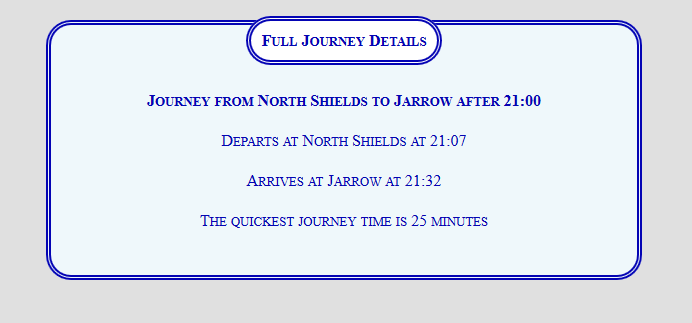
**Fig 28 - Selecting journey from Jarrow to North Shields at 18:00pm and return at 21:00pm**

****

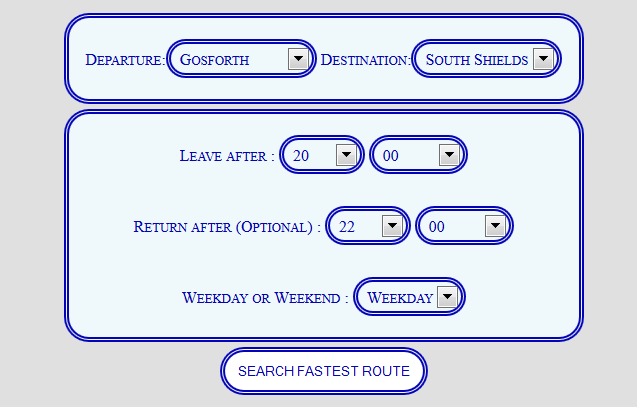
**Fig 29 – Departure journey for Jarrow to North Shields**

****

**Fig 30- Return Journey from North Shields to Jarrow**

****

**Fig 31- Selecting journey from Gosforth to South Shields at 20:00pm and return at 22:00pm**

****

**Fig 32 – Departing Journey from Gosforth to South Shields after 20:00pm**

****

**Fig 33 – Return Journey from South Shields to Gosforth after 22:00pm**

****

**NF2: Results can be obtained in less than 10 key strokes/clicks**

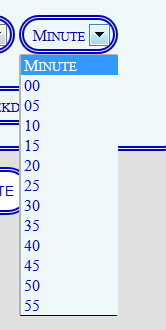
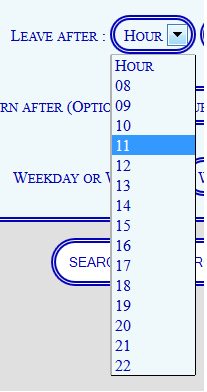
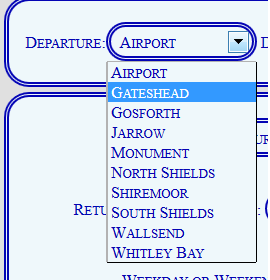
**Test Case: A search for a Weekend journey from Gosforth to Jarrow, at 11:00am, with a return journey at 14:00am**

**Expected Output: User can access results in less than 10 key strokes/clicks**

**Actual Output:**

To do this search, firstly the user has to choose the departure station from the drop down box, in this case, Gosforth. This is also done for the destination station. This represents two strokes or clicks. The user then selects an hour and minute to leave after. This represents a further two clicks or strokes. If the user chooses to do so they can select a return journey. This mirrors the ‘Leave After’ process of selecting an hour and minute. This is another two clicks. The user then selects whether he or she wants to make the journey on a weekday or weekend. After this the user presses the ‘Search Fastest Route’ journey and then is taken to the results page. Therefore the maximum number of clicks to get to the results page is 8 and the minimum (minus a return journey) is 6. Therefore this is evidence of the system allowing the user to get to the results page in less than 10 clicks/strokes. Therefore the system meets this requirement. These actions are shown in figure 34

**Figs 34 – Shows the various drop boxes associated with each click.**

****

****

**NF3: The system is available on most commonly used browsers**

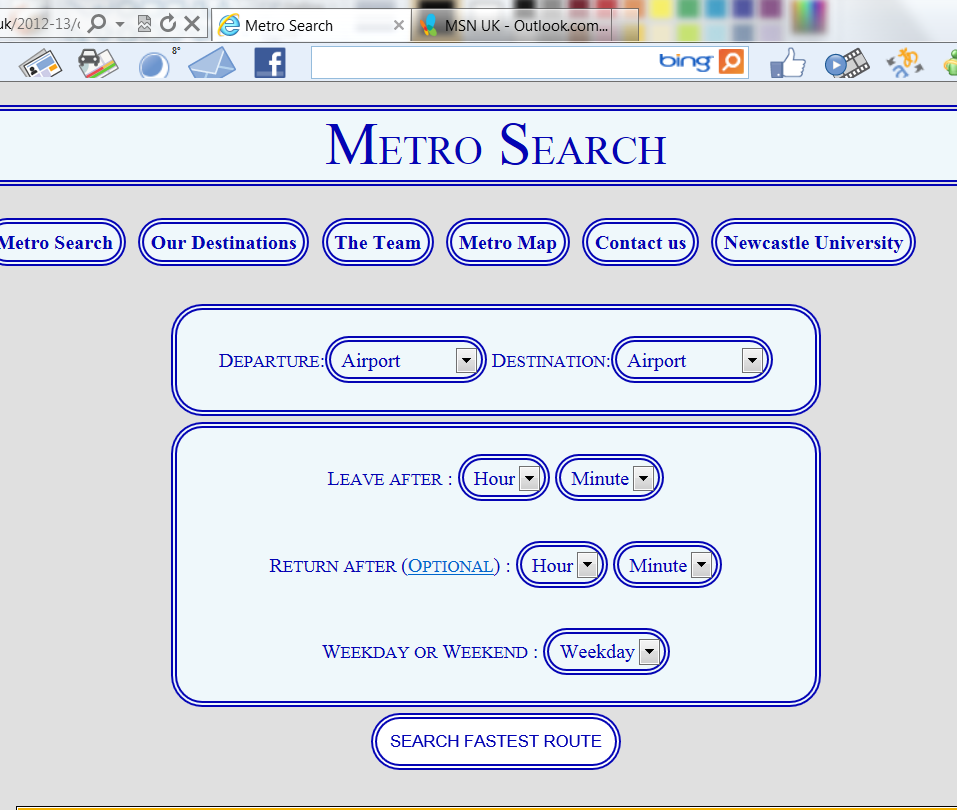
**Test Case: The website shown on ‘Firefox’, ‘IE’, ‘Chrome’ and ‘Safari’**

**Expected Output: website runs without any problems in all mentioned browsers**

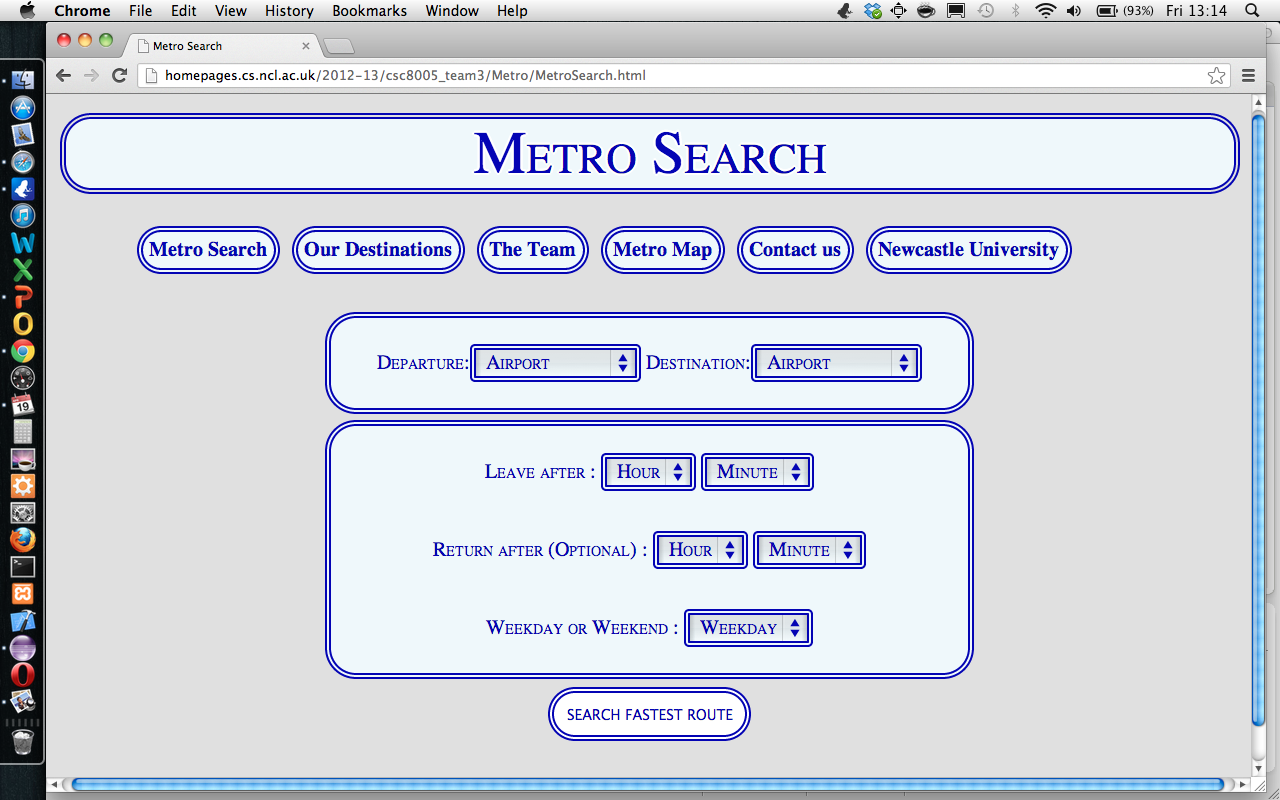
**Actual Output:**

All of the previous screenshots were taken whilst browsing using ‘Firefox’, figure 35 shows the system on Internet Explorer, figure 36 shows the system on Chrome and figure 37 shows the system on Safari. Therefore this is evidence of the system being available on the common browsers, meaning the system has passed in meeting this requirement.

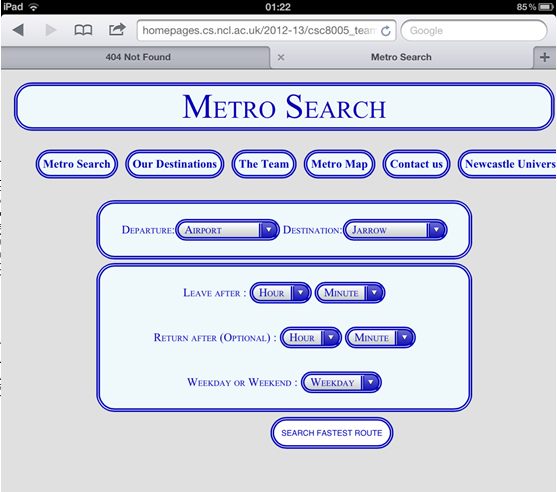
**Fig 35 – Website displayed in Internet Explorer**

****

**Fig 36 – Website displayed in Chrome**

****

**Fig 37 – Website displayed in Safari**

****

**NF4: System displays forgiveness to the user**

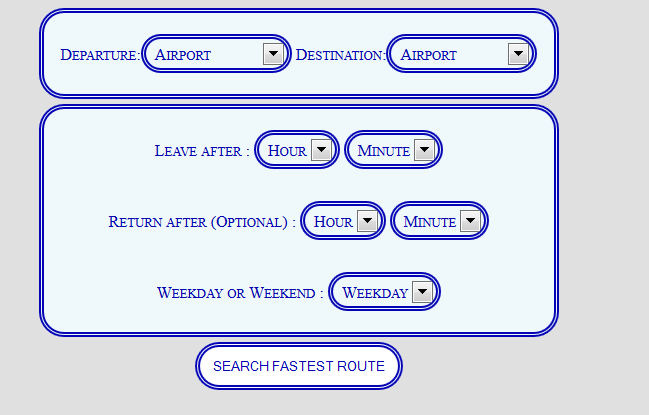
**Test Case: User inputs the same destination station as departure station**

**Expected Output: System gives error message**

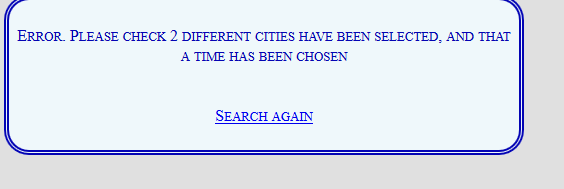
**Actual Output:**

If the user inputs the same destination as departure time, or fails to choose a time from the drop down box (as shown in figure 38) then the user is given this message on the results page (shown in figure 39). He or she is then given the option to search again, which again brings up the defualt website page. This therefore evidences forgiveness and therefore helps the system meet this requirement.

**Fig 38 – Figure showing incorrect input data**



**Fig 39 – Showing system response**

****

**NF5: Website can handle multiple users at any one time**

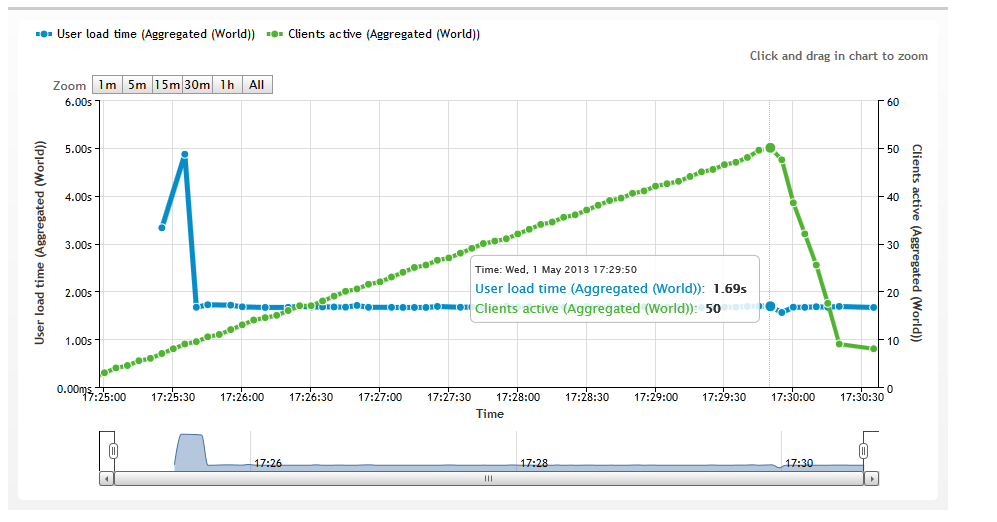
**Test Case:**The **w**ebsite was tested with<http://loadimpact.com/> which creates 50 client requests to the website over 5 minutes.

**Expected Output: Website is able to handle multiple requests from multiple users over a sustained period**

**Actual output:**

Figure 40 shows an initial spike to 5 seconds per load time, which quickly reduces to an average of 1.7 seconds and sustains this time throughout the test. This means that around 50 users could use the website without a significant load time delay. This evidenced that system can handle multiple users and therefore helps it meet this non-functional requirment.

**Fig 40- Shows the stress test**



**External User Testing**

To gather feedback from individuals not linked to the team we undertook side by side testing with 5 people to obtain data on the user experience. As this feedback can be influenced by the interviewer, 5 more people were asked to complete a closed questionnaire, anonymously rating the website. To avoid confusion, it was explained to each participant beforehand that the Metro Map used is different than real life version.

**Limitations**

The testing was carried out with a similar demographic, male, 23-28 years old, and computer literate. A test involving a wider spectrum would be necessary to present an accurate assessment of the usability of the site for the population, but we did not have time to undertake this.

Side by side testing

|  |  |
| --- | --- |
| **Demographic** | **Comments** |
| Male, age 25 | Said the site was easy to understand, it was clear what each function was and how to search a journey. Liked that the full journey details were hidden by the user by default, though found the drop down button hard to find at first. |
| Male, age 24 | Found the site straight forward. Did not provide any strong positive or negative comments, called the website “basic”. |
| Male, age 24 | User found the buttons on the site slightly confusing, as he did not know which ones to click. Once he had been instructed liked the site, especially the collapsible search results |
| Male, age 27 | Found the site fine, did not like the drop down options for stations as he thought typing the station names would be faster. |
| Male, age 26 | Found the site easy to use and gave positive feedback. Commented that the error message was not clear when forgetting to input a leave after time |

A problem identified was that users were unclear if the buttons on the page were clickable. The code was changed to highlight in a dark shade of blue when the user hovered over each button. Another comment was that with the text that we had used to display the full journey details “Full Journey Details” it was not apparent that this was a collapsible box concealing more journey details. Figure 41 shows the original display, and figure 42 how we changed it to provide the visual feedback.

The error message when a user forgets to input a time was mentioned as being unclear as to whether a user would need to include a return time as well. The wording was changed, as seen in figure 43 and figure 44.

The comment received about the station names being contained within drop down boxes rather than typed was consistent with our own findings, but for reasons mentioned in the technical summary we did not change this.

Overall the feedback received was extremely positive, all users commented that the website looked clean, and found the interface very easy to use. Aside from the buttons, when asked if they thought any part of the website was confusing, they all replied no.



Figure 41 – Unclear that ‘Full Journey Details’ was a button containing extra information



Figure 42 – Label amended to display ‘Click For Full Journey Details’, and to turn blue when hovered over

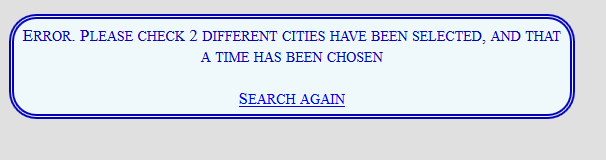


Figure 43 - Original message did not specify which time needed to be inputted

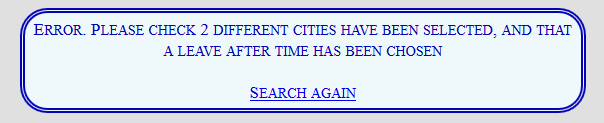


Figure 44 – Message amended to make clear it is the leave time that needs to be inputted

As the external testing was carried out a late stage, the screenshots in other parts of the testing section do not contain the amended text.

**External Survey results**

The survey was undertaken after the amendments had been made from the side by side testing. Each question, and the results the participants gave, are outlined below.

**Question 1** – Go onto the website and try to find a return journey from the Airport to South Shields leaving after 10:00, returning at 17:00. Then try to do a search for a single journey from Gateshead to the Monument leaving after 18:00. With 1 being complicated and 4 being very easy, rate how easy you found the interface to use.

|  |  |
| --- | --- |
| Option | Selected |
| 1 -Complicated | 0 |
| 2 – Not easy | 0 |
| 3 – Easy | 1 |
| 4 – Very Easy | 4 |

Note – 1 user picked easy, 4 users picked very easy.

**Question 2** – Attempt to search a journey from Airport to S Shields at 22.55 on the weekend. With 1 being complicated and 4 being very easy, rate how clear the message displayed explained the situation.

|  |  |
| --- | --- |
| Option | Selected |
| 1 -Complicated | 0 |
| 2 – Not easy | 0 |
| 3 – Easy | 0 |
| 4 – Very Easy | 5 |

Note – Message displayed says there are no trains after this time.

Question 3 – Attempt to find which stations the user would have to change at from South Shields to the Airport, leaving at 10.00, returning at 19.00. With 1 being complicated, and 4 being very easy, rate how easy this was to find.

|  |  |
| --- | --- |
| Option | Selected |
| 1 -Complicated | 0 |
| 2 – Not easy | 0 |
| 3 – Easy | 0 |
| 4 – Very Easy | 5 |

Question 4 – The website contains additional information on each station in the network. Attempt to find the extra information on Whitley Bay. With 1 being complicated, and 4 being very easy, rate how easy this is to find.

|  |  |
| --- | --- |
| Option | Selected |
| 1 -Complicated | 0 |
| 2 – Not easy | 0 |
| 3 – Easy | 1 |
| 4 – Very Easy | 4 |

The results were positive with nearly all users finding the website very easy to use. Of particular interest was question 3, as this was the task (finding full journey details) which users had found confusing during the side by side training, so we were pleased that the amendments made to the display had been successful.

# APPENDIX D – CODE SUMMARY

Project Name: *Metro Train Timetable*

Team Number 3

Document Information – Summary of PHP Code

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | | |
| Project Name: | Metro Train Timetable | | |
| Prepared By: | Gary Carr, Chris Kerr, Kefu Li, Jasjot Mattu, Antao Xu | Preparation Date:28/4/2012 |  |
| Email / Phone: | [g.carr@newcastle.ac.uk](mailto:g.carr@newcastle.ac.uk) |  |  |
| Document Version No: | 1.1 | Document Version Date: 02/5/2013 |  |
|  |  |  |  |

Version History

| Ver. No. | Ver. Date | Revised By | Description |
| --- | --- | --- | --- |
| 1.0 | 28/4/2013 | Gary Carr | Document compiled |
| 1.1 | 1/5/2013 | Chris Kerr | Addition of HTML and CSS |

Summary of PHP Code

The full commented PHP code has been submitted with the group report. This section contains a summary of the files submitted and a description of its functionality. There is no hard coding within the PHP; the database can be updated without having to change the PHP code.

The home page is the file index.html. The file Style2.css links to every other file to set the display for each page.

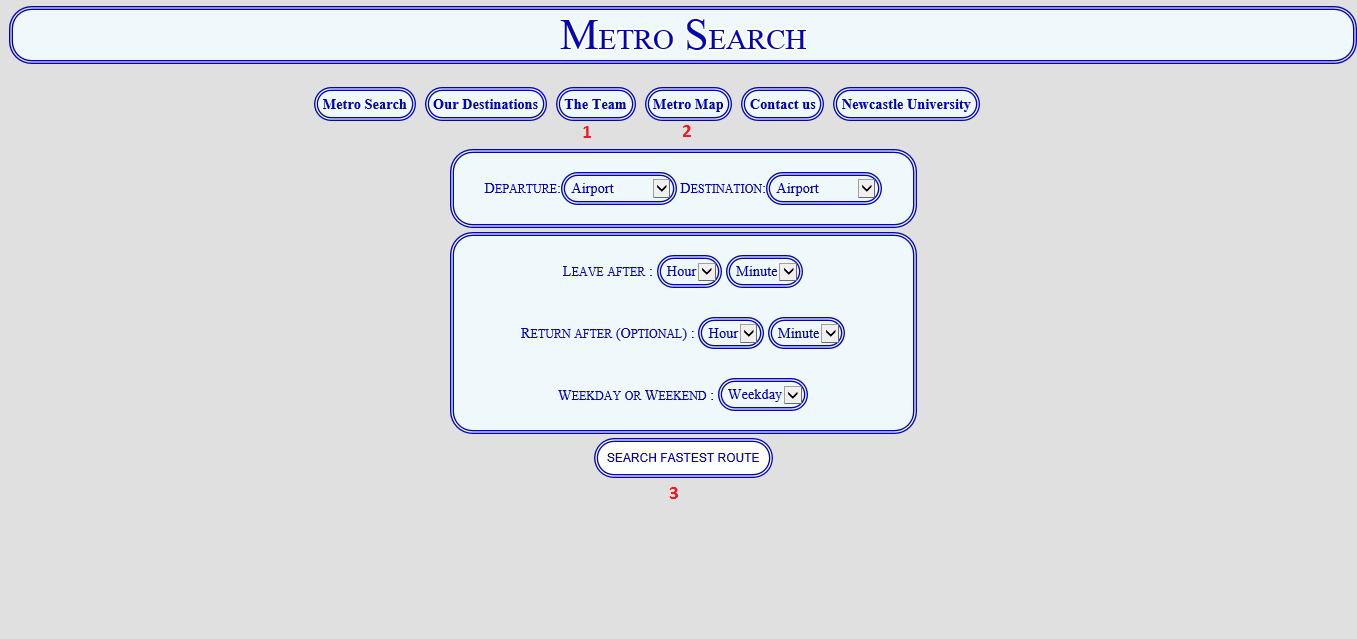


Figure 1 - Home page

Below is a list of other files used in the website. The numbering on figure 1 illustrates where the files are linked to from the home page.

1. Link to pictures of each group member

* Antao.html
* Antao.png
* Chris.html
* Chris.jpg
* Gary.html
* Gary.jpg
* Jazz.html
* Jazz.jpg
* Kefu.html
* Kefu.jpeg

1. Map of the metro station

* Map.html
* Metro\_route.png

1. Finding fastest route

* dijAlgorithm.php
* generalFunctions.php
* MetroSearchResults.php
* RouteFinder.php

Each page on the site contains the same 6 header buttons (Metro Search, Our Destinations etc.) to keep consistency throughout.

For finding the fastest route (the files in group 3), the home page (index.html) posts the drop down variables inputted by the user to MetroSearchResults.php. In its first version the results page contained several hundred lines of code in one method. As the coding evolved the code was broken down first into separate functions and then put into different PHP files completely. This segmentation made the code easier to read and develop.

Below is a summary of the functions contained in the PHP.

MetroSearchResults.php – The main PHP page, this included the other PHP pages, calls the functions, and passes the variables to the next function. This page determines if there is a return journey, and if so swaps the destination and departure stations and runs the PHP code for a second time.

generalFunctions.php –

* assignVariables() – Stores the variables inputted by the user
* connectDB() – Creates a connection to the MySQL database
* selectAllPossibleRoutes() – Selects all possible trains routes and returns them as array
* displayResults() – Prints out the summary of the users journey

dijAlgorithm.php – This class file was created by Rick Purple and found at <http://nirapath.googlecode.com/svn-history/r47/trunk/php/scripts/function/sortDijkstra.php>. The class allows the input of a start and end node, with a minute cost and returns the fastest result. This code has not been changed, and is the only code not written by the group.

RouteFinder.php –

* determineStationsBetweenRoutes() – Takes all possible routes outputted by selectAllPossibleRoutes(). Every route which has the correct start and end destination enters a for- each loop at line 29. Multiple arrays can enter the loop and the fastest is stored at line 72. That array is then returned.
* outputFastestRoute() - Takes the fastest array found from determineStationsBetweenRoutes(). Calls other functions to print the journey details and returns the times taken.
* delayCost() – Determines if there is a delay between a train arriving at a station and the next train that a user needs to get on. The delay time is then outputted. If a user does not have to change trains, the user is told to stay on the train.
* moreThanTwoStationsFound() – This function takes a starting and next station and is called if there are more than two stations left on the users journey. For instance if the user was travelling from Airport->Gosforth->Monument->Gateshead, the function would be called with the parameters (Airport, Gosforth), (Gosforth, Monument). The journey between those 2 stations is then determined.
* finalTwoStations() – This function takes a starting and next station and is called if there are only two stations left on the users journey. For instance if the user was travelling from Airport->Gosforth->Monument->Gateshead, the function would only be called at the end with parameters (Monument, Gateshead). The journey between those 2 stations is then determined.
* noRouteFound() – If no route is possible for the journey (as the user wants to leave the station at too late a time) then this function is called to display a message.
* extra\_journey\_details() – Prints the journey details between each station

MySQL Database

The relational database which holds the train networks was built using MySQL and is queried using MySQL commands by the PHP code.

The database was designed to be as simple as possible for a user to understand the data and the meaning of the column names. The database is in Boyce-Codd normal form to ensure no duplication, and that the integrity of the database is held when updating or deleting data.

The database contains 4 tables

* final\_metro\_trainstations
* final\_metro\_costs
* final\_metro\_week\_dep/final\_metro\_weekend\_dep

final\_metro\_trainstations– Contains a list of the station names on the train network.

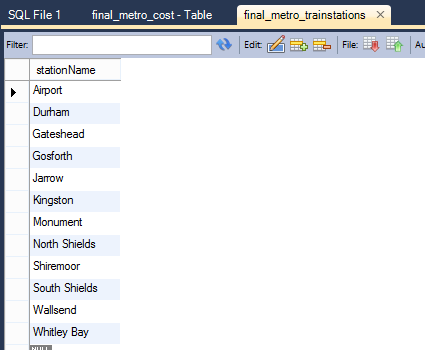


Figure 2 - List of station names

final\_metro\_cost – Contains a primary key (pathID) which is the unique name for the route. The fromStation and toStation are foreign keys of final\_metro\_trainstations. To ensure integrity, the database has been set so that if a station is deleted from final\_metro\_trainstations, then it is cascaded to delete any entry containing that station in this table.

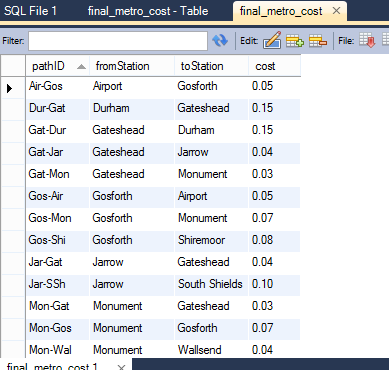
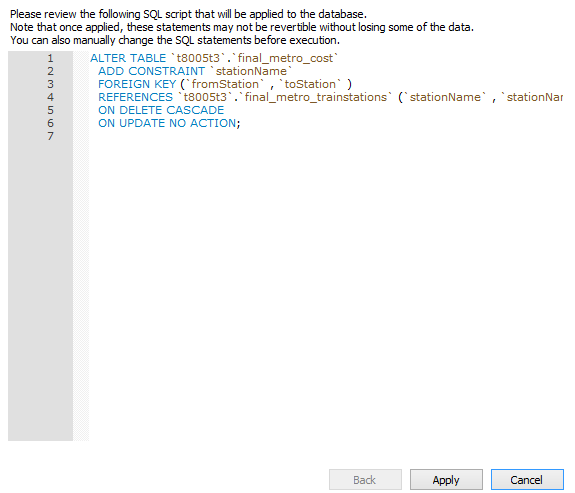
 

Figure 3 - Cost between stations Figure 4 – Setting up foreign key

The cost is the minutes between each station. 2 minutes of the cost is the time the train stops between each station. So for the route from the Airport to Gosforth, the journey takes 3 minutes and then the train stops for 2 minutes before moving on.

Each station can only see its immediate neighbour. It is the PHP code which then creates the total map.

final\_metro\_week\_dep and final\_metro\_weekend\_dep

Both tables have an identical structure. The only difference is that the weekend timetable runs for fewer hours than the weekday timetable.

The table contains a primary key (depNo) to identify the journey, and a foreign key (pathID) from final\_metro\_cost. Again using a foreign key ensures data integrity throughout the database. The column leavesAt contains the time the train leaves the starting station.

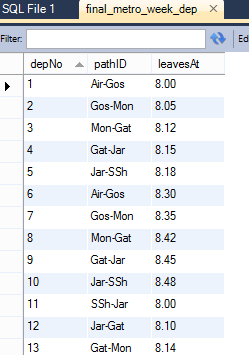


Figure 5 - Time each train departs

Adding a new station to the end of the line

A template has been provided to enter a new station into the database, as we did not want the client to have to manually input each entry into the MySQL database. To keep it simple for an inexperienced user we grouped the majority of the variables at the top allowing for quick amendment, as shown in figure 4.

Figure 4 is an example of adding a new station (Durham) to the network connected to Gateshead. Two new path ID’s are created, the minutes between the stations set, and the number of hours the train would run for.

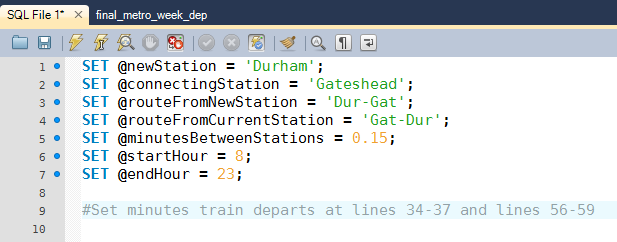


Figure 6 - Setting a new station

Finally the user has to enter the minutes that the train leaves the station. This is shown in figure 5 at line 34, where the user has stipulated the train will leave on the hour, and half past the hour.



Figure 7 - Entering the minutes leaving the station

In this example the user only has to change 11 variables to create a new station, which will generate 60 new departure times between the stations. The entirety of the template has been submitted with the group report, in the file newStation.sql.

Note - If the train ran at a different time every hour then this script would not be suitable. In that scenario the code at figure 5 would need to be amended to enter each leaving hour and minute rather than using the while loop to increment.

Removing a station

To remove a station from the network, the user simply has to enter the name of the station into the code in figure 6. The database tables are linked by a foreign key so the delete will cascade to remove ‘Airport’ references from the other tables, ensuring data integrity.

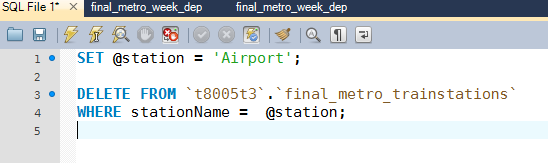


Figure 8 - Removing a station

Adding a new station in the middle of the network



Figure 9 - New station Pelaw

If a user wants to add a new station to the middle of the network e.g. adding Pelaw between Jarrow and South Shields, then the following SQL commands are used.

First the existing link between Jarrow and South Shields is removed. Cascading will also delete any existing departure times between the stations.

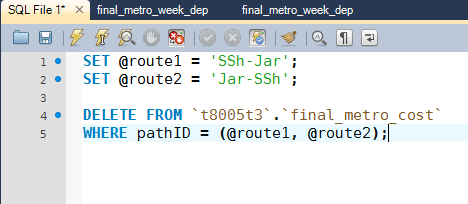


Figure 10 - Removing existing links between stations

Then creating Pelaw is treated the same as creating a new station at the end of the line. A link can be created first from Pelaw to South Shields.

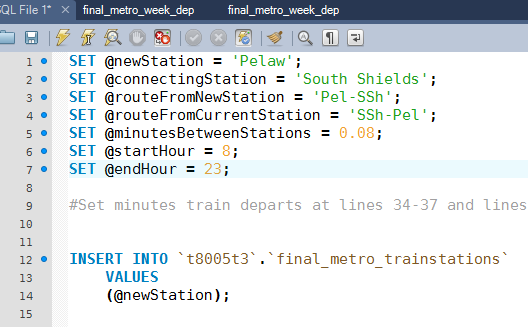


Figure 11 - Adding Pelaw to the network

Finally to connect Pelaw to Jarrow, code has to be commented out as Pelaw does not need to be added as a new station again. The code is shown in figure 10, with lines 12-14 commented out.

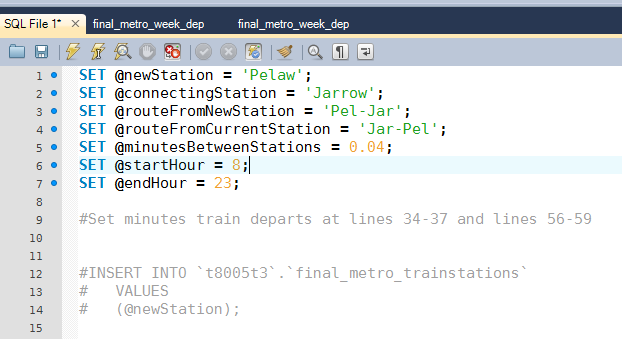
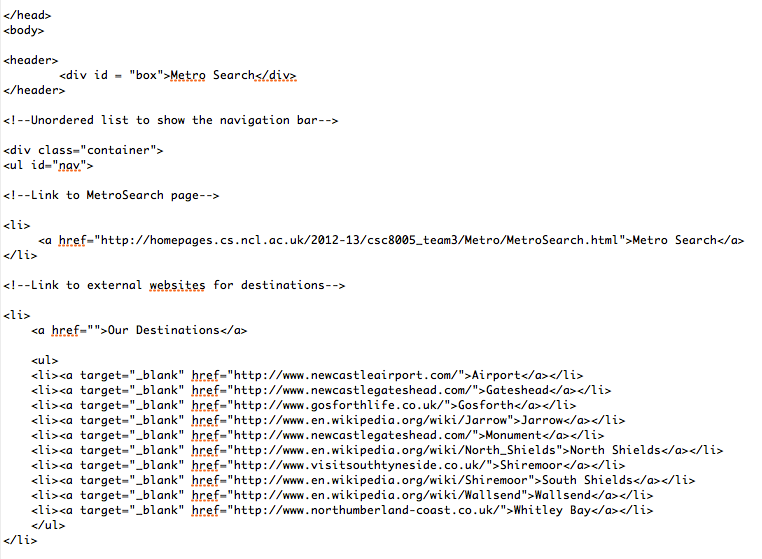


Figure 12 - Adding Pelaw to Jarrow

The new station will now be added to the network. The PHP code does not need to be amended; it will automatically find the new station and calculate the new route cost.

Summary of HTML Code

The full commented HTML code has been submitted with the group report. This section contains a summary of the files submitted and a description of its functionality. The HTML is written to HTML5 standard and aimed at the latest browser updates.



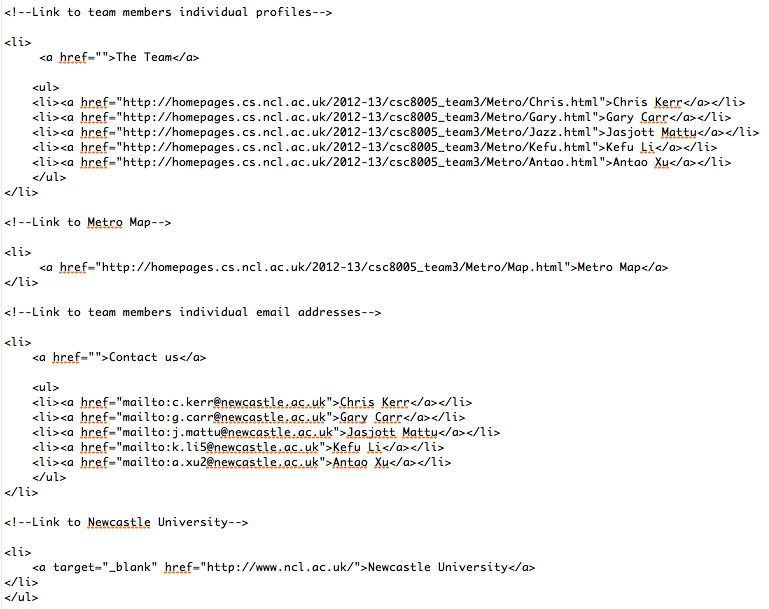
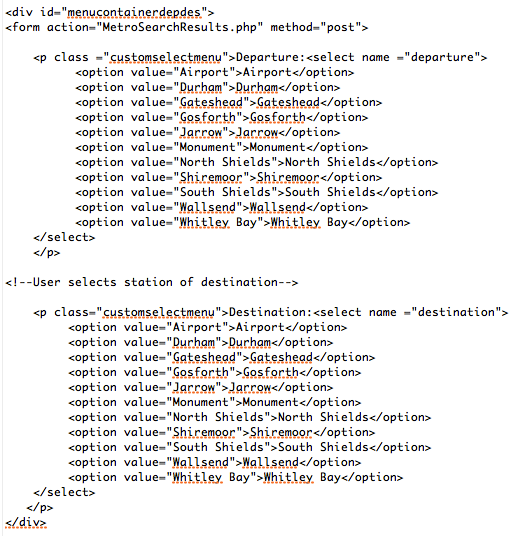


Figure 13. HTML for heading and navigation bar

This section of HTML remained a constant for all of the webpages. It provides links to various internal and external webpages.



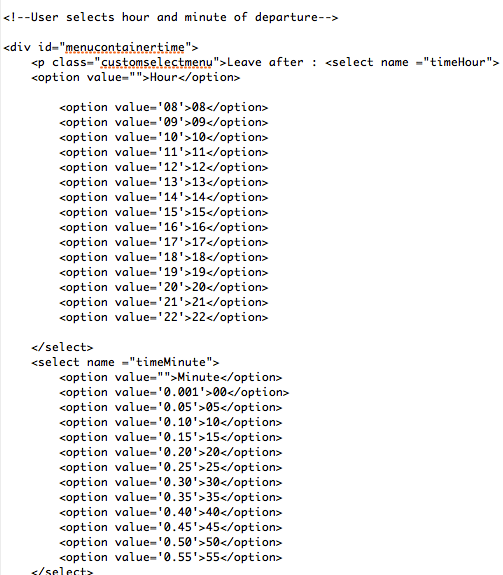


Figure 14. Selecting a station of Departure and Destination and the desired leaving time

The HTML displays the destinations available for the user and the leaving times in five-minute increments. The PHP then returns a train time applicable to this situation.

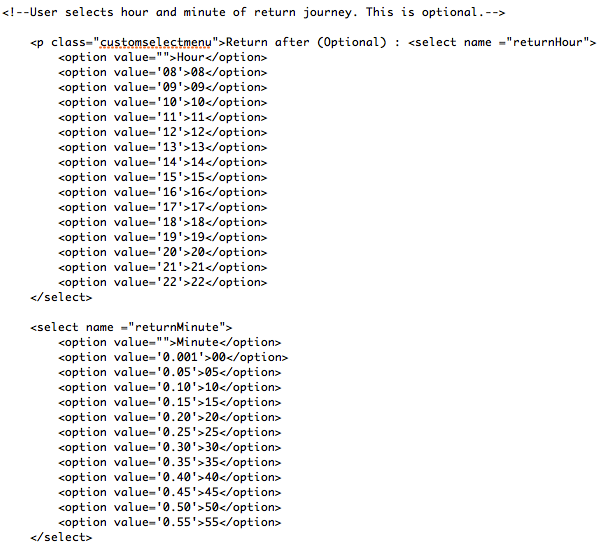


Figure 15. HTML Code for a return journey

The webpage offers an optional return journey for the user. This HTML code is similar to the outbound journey.

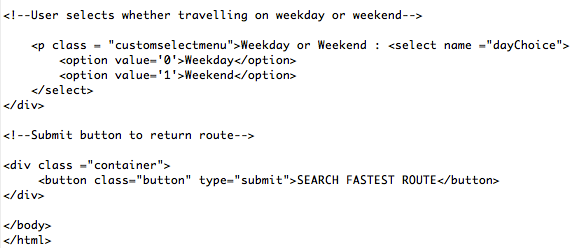


Figure 16. HTML Code for the weekday and weekend option and the submit button

The user is given the option to enter a weekday or weekend, as different schedules exist for each. The submit button enters the given user information to the PHP file to return the correct results.

HTML Validation

The HTML was tested for compliance with HTML5 standards on the W3C website http://validator.w3.org/, and passed with no errors found (figure 15). All HTML pages on the site have been successfully tested.

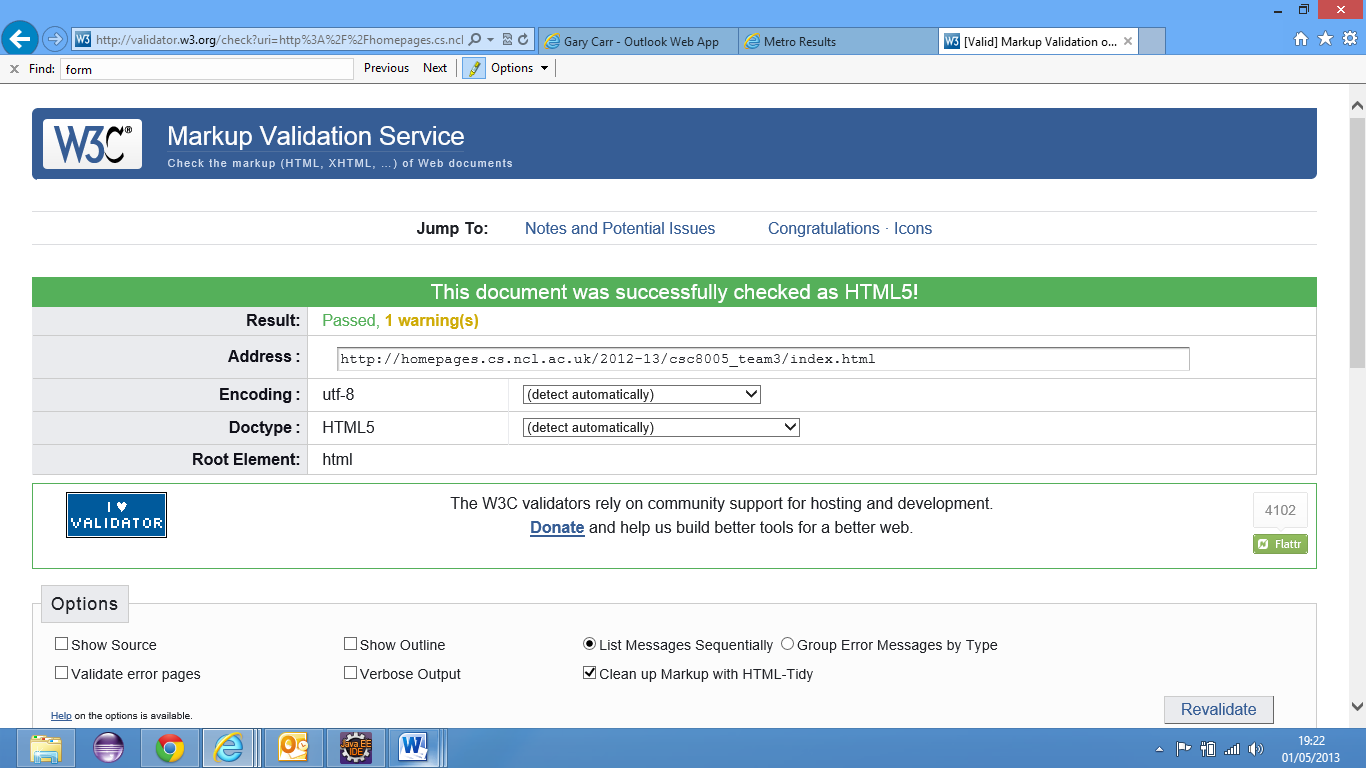


Figure 17. Website passed compliance test for HTML5

Summary of CSS code

The full commented CSS code has been submitted with the group report. This section contains an example of the submitted. The CSS is written to CSS3 standard and aimed at the latest browser updates and to use CSS tags.

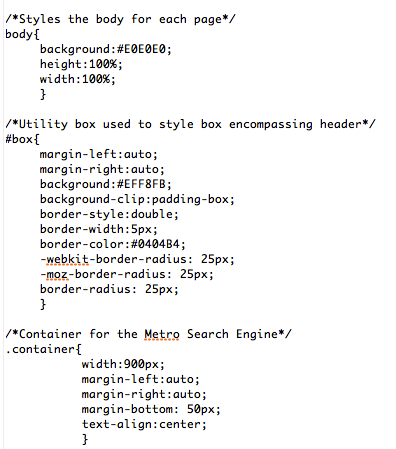


Figure 18. Sample of CSS code

The CSS uses CSS3 and web-kits to style the HTML. The different browsers have different results when the same CSS is used. These web-kits reduce this difference. The CSS is commented to provide viewers with information as to their usage.

CSS3 Validation

The CSS was tested for compliance on the W 3C website <http://jigsaw.w3.org/css-validator/>, and passed with no errors found (see figure 16).

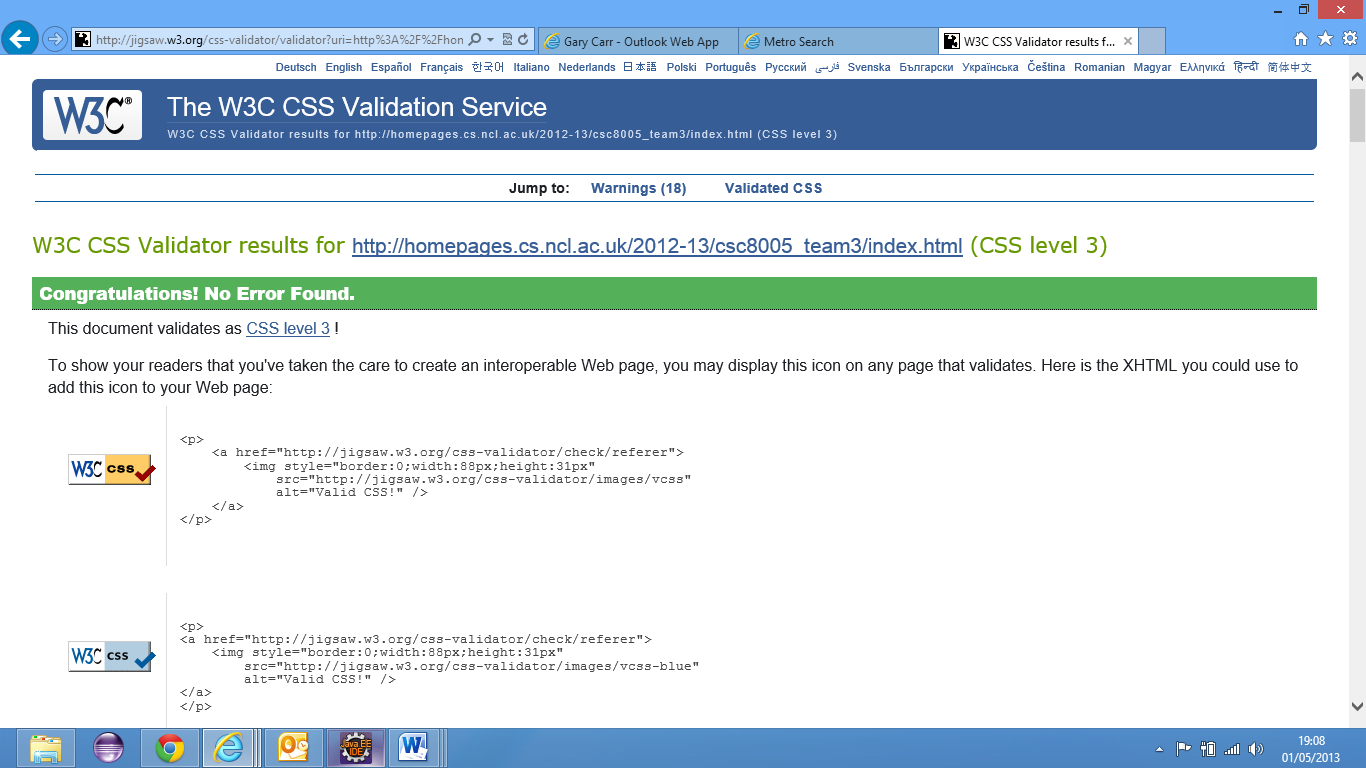


Figure 19 CSS passing the compliance test

# APPENDIX E – MINUTES

## Meeting 1

MEETING HELD: 15/11/2012

MINUTES

VENUE: Robinson Library room 432

TIME: 10am

CHAIRMAN: Gary Carr

SECRETARY: Christopher Kerr

PRESENT:

Gary Carr

Christopher Kerr

Kefu Li

Jasjot Mattu

Antao Xu

John Nicholson (monitor)

APOLOGIES: N/A

TASKS ARRIVING FROM PREVIOUS MEETING: 1st meeting of the group

GENERAL BUSINESS:

* Discussion as to what resources we need, specifically mentioned resources such as JQuery and W3 Schools training website.
* Discussion at functional requirements to decide what is high, medium and low priority
* Finalized these within the specification document, Task carried out by Christopher Kerr
* Detailed discussion of what website browsers to use for railway website.
* Discussed and formalized assumptions we are going to build the software on.
* Discussed initial thoughts on website design

ALLOCATED TASKS TO BE COMPLETED:

* Set up a team Google doc account
* Ask Steve questions on Monday – buying tickets online?
* Team to look at technology to be used to on different parts of project, bring to next meeting.
* Team to think of questions to bring to Steve on Monday.
* Complete agenda prior to each formal meeting.

NEXT MEETING: 10.30am 27th November

MEETING CLOSED: 11am

## Meeting 2

MEETING HELD: 27/11/2012

MINUTES

VENUE: Robinson Library room 432

TIME: 10am

CHAIRMAN: Christopher Kerr

SECRETARY: Jasjot Mattu

PRESENT: Gary Carr

Christopher Kerr

Kefu Li

Jasjot Mattu

Antao Xu

John Nicholson (monitor)

APOLOGIES: N/A

TASKS ARRIVING FROM PREVIOUS MEETING: Completed Agenda for this meeting, team Google docs account, no requirements to be able to buy tickets.

GENERAL BUSINESS:

* Specification requirements due in on 14th Dec, 4 weeks till this is in – focus on the work required to complete this to a high level. Agreement initial document will be completed by the 7th so we can review it over the following week.
* Group member with highest individual elicitation score will finalize specification requirements. Members will all contribute to the project through draft write-ups.
* Review individual elicitations and bring together to gain an idea of where we are.
* Complete list of Functional requirements recorded by Jasjot.
* Complete list of non-functional requirements recorded by Jasjot.
* Discussion of breakdown of the roles in the group to complete the project. Agreement to leave this till more in known about the project.
* Agenda to be completed ahead of meetings and feedback to John prior to meetings.
* Agreement to next chairman (Chris) and secretary (Jasjot)

ALLOCATED TASKS TO BE COMPLETED:

* Ask Dr. Lindsay Marshall about browser sniffing, to help to determine what browsers to use for website, and hardware, software we require.
* Ask Chris Ritson about hardware requirements, general look at hardware requirements.
* Review “An introduction to requirements engineering” by Ian Bray.
* Team to think of use cases for our project.
* To begin writing and formalising specification document.
* Assigned tasks for specification document.

NEXT MEETING: 10am 11/12/12

MEETING CLOSED: 11am

## Meeting 3

MEETING HELD: 11/12/2012

MINUTES

VENUE: Robinson Library room 432

TIME: 10am

CHAIRMAN: Jasjot Mattu

SECRETARY: Antao Xu

PRESENT: Gary Carr

Christopher Kerr

Kefu Li

Jasjot Mattu

Antao Xu

John Nicholson (monitor)

APOLOGIES: N/A

TASKS ARRIVING FROM PREVIOUS MEETING: Feedback from Dr Marshall and Chris Ritson, demonstration of individual use cases, draft of specification documents.

GENERAL BUSINESS:

* Teammates explain their parts, others discuss, modify. John gives some necessary ideas. (We can keep submitting Requirements Specification).
* Kerr--finding fastest path, summarize.
* Jazz--scope (in/out).
* Chris--describe both of hardware platforms and software platforms (which browser supports).
* We already agreed functional and non-functional requirements few weeks ago, everyone is happy with constraints and dependencies. Gary explains this part.
* Henry--GUI, Kerr gives some ideas.
* Li-Use Cases: how user’s accesses functionality, explain the processing, remove unnecessary functions, and discuss the disability situations.
* Chris—definitions.
* Too much pages, we find out some methods to reduce pages.
* Everyone is happy with the existing final draft.
* Christmas task: SQL; Database; JS; PHP; what should we do; what could we do.
* Keep contacting during Christmas by email.
* Discuss how to arrange works.
* Any other points about project.
* Summarize of group project so far.
* John explains Gantt Chart Anatomy; we could create a Gantt chart by Excel.

ALLOCATED TASKS TO BE COMPLETED:

* To be completed: W3Schools, learn to create website; An Introduction of requirements engineering.

NEXT MEETING: 10am 04/02/13

MEETING CLOSED: 11am

## Meeting 4

MEETING HELD: 04/02/2012

MINUTES

VENUE: Robinson Library room 432

TIME: 12noon

CHAIRMAN: Antao Xu

SECRETARY: Kefu Li

PRESENT:

Gary Carr

Christopher Kerr

Kefu Li

Antao Xu

John Nicholson (monitor)

APOLOGIES: Jasjot Mattu

TASKS ARRIVING FROM PREVIOUS MEETING: HTML, CSS, PHP, MySQL tutorials.

GENERAL BUSINESS:

* Discuss the problems on previous assumptions and plan on next stage work using JavaScript, PHP, Ajax, MySQL, etc. Make it 15 stations for the project and talk about how to find a shortest way.
* Make it clear on the concept of reliability, Usability and referred XML; Login section is optional.
* John demonstrates some basic information on PHP (PHP is case sensitive). Use Dreamweaver to exploit web pages, but it sometimes have error code problems.
* John claims the difference between HTML4 and HTML5.

ALLOCATED TASKS TO BE COMPLETED:

* Be able to build diagrams, think about how data can be defined in the database, how to construct web pages and communicate with the database.

NEXT MEETING: 12noon 05/03/13

MEETING CLOSED: 1pm

## Meeting 5

MEETING HELD: 05/03/2012

MINUTES

VENUE: Robinson Library room 432

TIME: 12noon

CHAIRMAN: Gary Carr

SECRETARY: Christopher Kerr

PRESENT: Gary Carr

Christopher Kerr

Kefu Li

Jasjot Mattu

Antao Xu

John Nicholson (monitor)

APOLOGIES: N/A

TASKS ARRIVING FROM PREVIOUS MEETING: Draft website and functionality, test, new design layouts.

Gary – PHP, Requirements Elicitation

Chris – HTML, CSS, Design Brief

Jas – Dijkstra’s algorithm

Antao & Kefu – SQL, Database

GENERAL BUSINESS:

* Gary reiterated what has occurred in our informal meetings to John to update to the current situation.
* Kefu away over Easter till 3rd April, Antao till 6th April, Jas in London yet not on holiday.
* Gary spoke about what he did; placing the database online instructions, attempting to join the PHP to the database, Chris and Gary spoke about the documents needed for the assignment. Placing a methodology into the team report.
* Jas mentioned recording how we tested the software, using other groups as the test subjects.
* Antao and Kefu have placed the database online
* Chris looked into wrapping website onto a mobile device, told by John that is too far outside scope. Completed a design brief and everyone else will send the parts of the design brief they completed to Chris to add in.
* John said we all have to be involved in the team presentation.
* Group allocation of marks mentioned.
* Jas spoke about working on Dijkstra’s algorithm and how far he got with the work. 2(n-2) used to work out the stopping time in between stations.
* Antao and Kefu have created 3 stations in the database as a tester will create 10 in total.
* Chris to take all meeting notes and place into the format.
* Antao and Kefu to send screenshot of database to Jas.

ALLOCATED TASKS TO BE COMPLETED:

* Speak to Steve about the methodology
* Leave obvious gaps in the presentation.
* How long is presentation? Ask Steve.
* Continue to complete the tasks we set in the informal meeting – Gary doing the methodology as an additional test, Chris to do CSS bring in everything everyone has contributed.

NEXT MEETING: week beginning 15th April

MEETING CLOSED: 1pm

## Meeting 6

MEETING HELD: 23/04/2012

MINUTES

VENUE: DAYSH.Rack.Side.PC

TIME: 12:10 pm

CHAIRMAN: Christopher Kerr

SECRETARY: Jasjot Mattu

PRESENT: Gary Carr

Christopher Kerr

Kefu Li

Jasjot Mattu

Antao Xu

John Nicholson (monitor)

APOLOGIES: N/A

TASKS ARRIVING FROM PREVIOUS MEETING:

* Speak to Steve about the methodology
* Leave obvious gaps in the presentation.
* How long is presentation? Ask Steve.
* Continue to complete the tasks we set in the informal meeting – Gary doing the methodology as an additional test, Chris to do CSS bring in everything everyone has contributed.

GENERAL BUSINESS:

* Showcase finalized website to John
* Chris and Gary spoke about adding a print page button to the website. However John suggested that as long as journey times could be seen printing through browser would be fine. All agreed that it should be left as printing page solely through browser.
* Chris spoke about problems with validating the HTML code.
* Jas mentioned the testing, how it was from a user point of view. John said this was fine.
* Gary talked about adding information features to website. All agreed that we were short for time to do this.
* Began discussion about the team presentation.
* Agreed on who should say what and on what order.
* Jas to begin with group dynamics went on to discuss ideas he had for this in front of the rest of the group.
* Antao talked about the waterfall model , said aloud his ideas for this
* Kefu discussed his ideas for talking about problems during the design review as a part of the presentation
* Chris talked about his ideas for the HTML and website experience for the presentation
* Gary showcased his ideas for showing database functionality
* Agreed on presentation order; Jas, Antao, Kefu, Chris and then Gary
* Agreed to do practices runs during informal meetings before the presentation.
* Talked about the design review, allocated who should do what, namely Gary- Overview, Chris – GUI, Antao- Overview of components, Kefu – Class Diagrams, Jas- Activity and Sequence diagrams.
* Agreed to ask Dr. Steve Riddle for further clarification about design review.
* Agreed to discuss group report and appendices in next informal meeting

ALLOCATED TASKS TO BE COMPLETED:

* All members to finalize their respective parts of the presentation
* To meet up and do a practice run of the presentation
* Do task with the design review
* Ask Dr. Steve Riddle for clarification of design review.
* Finalize Group report and Appendices in informal meetings.

NEXT MEETING: Final Formal Meeting

MEETING CLOSED: 1:10pm

1. Eastcoast Mainline website found at http://www.eastcoast.co.uk/ [↑](#footnote-ref-1)
2. National rail website found at http://www.nationalrail.co.uk/ [↑](#footnote-ref-2)
3. ‘http://en.wikipedia.org/wiki/Scope\_%28project\_management%29’ [↑](#footnote-ref-3)
4. http://gs.statcounter.com/#browser-ww-monthly-200807-201303 [↑](#footnote-ref-4)
5. http://gs.statcounter.com/#browser-ww-monthly-200807-201303 [↑](#footnote-ref-5)