**Technical Summary**

This section is a high level summary of the coding used in the website. A detailed description can be found in appendix D. Team 3’s website can be found at <http://homepages.cs.ncl.ac.uk/2012-13/csc8005_team3/>. The database is held at <https://svn.cs.ncl.ac.uk/viewvc/csc8005_team3>.

Website display

HTML5 and CSS 3.0 were used for the display of the website as they are the latest standard in web development. Each part of the site was designed to apply good principles of human computer interaction, in particular against the 10 Nielsen heuristics of design (see Appendix B). For this reason the results page uses a JavaScript function to hide the results of the journey search within a collapsible container, minimizing the data outputted to the console. All files have been tested and are compliant with the online W3C Mark-up Validation Service for HTML5 and CSS3.

Database

A relational database was created in MySQL and consists of 4 tables in Boyce-Codd form; the first table is a list of station names, second states each stations immediate neighbour and the minutes taken to travel between those stations, and the third and fourth lists the departure times every train leaves every station at the weekday, and the weekend. Primary and foreign keys link every table to ensure no duplication and consistency. Template scripts have been provided so that a client can add or remove stations or routes whilst only having to change less than 25 words or numbers.

PHP

To return the results of the search it was first considered to use Java, as the group had most familiarity with that language, however it was later decided that PHP would be more appropriate as it is would not require any plug in’s for web browsers. The PHP takes the user inputted variables from the home HTML page and builds an array from the MySQL database of all possible routes on the network, and passes these into a Dijkstra’s algorithm class, which returns the fastest journey. Dijkstra’s algorithm was found online and is the only part of the code not developed by the team. The source is stated in appendix D pg.53.

The PHP takes into account delay time at the stations and outputs if the user has to stay on the same train, or get off at the station and wait. The times are outputted to the user in the results page. If the user has inputted a leave time later than the trains are running, a message is displayed informing them.

The code allows for an optional return journey. In this instance the PHP executes once, then swaps the departure and destination variables, updates the leave time and executes again. The results page is displayed differently if 2 journeys are processed.

**Testing**

Testing for the system was done internally from a user point of view, and is explained in detail in Appendix C. Testing followed the black-box model of testing as it looked at the system as a whole, and seeing what functionality the system could do. Doing it from this point of view, not only gave perspective as to how the user would interact with the system, but provided an opportunity to showcase from the user point of view as to whether the system had passed or failed in matching the functional and non-functional requirements.

Testing was undertaken at every part of the development process, ensuring new code was returning the results set out in the original specification (see Appendix A) before integrating it into the working system. This also was the basis for various prototypes of the code and early versions of the system. The code was tested, the results were looked at and then modifications we made as appropriate. The version tested from the user point of view is the agreed final design, but numerous iterations were developed to weed out potential bugs in the system.

Below is a table of the functional (marked with F) and non-functional (marked with NF) requirements that crucially could be tested by the system. The tables show whether the system has passed in meeting these requirements. Overall the system passed in matching all the requirements listed. This evidences that the system’s implementation fulfils its mission in providing the features the team set out to make. Of course there were some requirements that couldn’t accurately be tested or evidenced, such as whether the system was stable 99% of the time.

An extended breakdown of this can be found in appendix C underneath the internal testing headings, detailing expected and actual outputs as well showing screenshots of the system in action.

|  |  |  |
| --- | --- | --- |
| 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 |

|  |  |  |
| --- | --- | --- |
| 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 |

External testing was undertaken which allowed the team to change the website based on recommendations from actual users. The results of this are in Appendix C.

Possible enhancements

There are some areas where we felt that improvements could have been made to the site which were not implemented.

* If a user attempts to search a return journey where the time entered for return is too late, the display of the results page is inconsistent with the normal display style. This was discovered at too late a stage to fix.
* At an early stage our code simply returned the fastest route, and had no time table for trains. At this point the user could specify if they wanted to go via or avoid a certain station. However we made a choice that it was more important (and realistic) to implement a full working timetable into the PHP and database, and incorporate waiting times at stations. Ultimately this made the code became too complex to avoid certain stations, so the feature was dropped.
* Originally a JQuery script was to be used to list the available stations rather than drop down boxes, which would be more efficient if the train network was larger. As the network only contains 10 stations, and due to a lack of time to test, we did not implement this.
* The requirement to add and remove new stations is implemented by changing variables strings and numbers within a MySQL script template. At a late stage we considered designing a GUI interface to simplify this, however the vast amount of scenarios for a train timetable was likely to make the GUI more complex than entering through the original script. A future development would be to create the GUI.

Group work

At the start of the project when our coding skills were weak the team was split into two groups to allow concentration on developing specific parts of project, the PHP and the MySQL database. By setting clear targets and maintaining constant communication we were able to merge the sections in early March into a working prototype, which evolved into the final version of the website. The structure of delegating tasks to individuals, or small groups, was maintained throughout and the quality of work was verified by repeated testing of each other’s code. This co-ordination allowed us to deliver the website in time, and to specification.

Team 3 has decided that each member deserves an equal weighting (20%) of the total mark of the project.