# Analysis

## Introduction

#### Background

The Tourist Information Centre in Cambridge provides information about local accommodation, public transport, conference services, events, UK holiday information, walking tours of Cambridge, maps & guidebooks and much more. They already have a website and software that run in their office that provide general information about local events and information. However they do not have a computerised version for information about public transport. Currently the Tourist Information Centre uses printed timetables provided by Stagecoach to store information about buses. In the past this system was sufficient as Cambridge had very few buses and it was easy find a way around. Over time Cambridge has grown and expanded and now there are over 30 buses coming from and to Cambridge. A few problems have arisen with the current system and Joe Johnson, who works at the information centre, has asked me to make a computerised solution to this problem.

#### Project Definition

Client: The Cambridge Tourist Information Centre

Contact: Joe Johnson

Peas Hill

Cambridge,

CB2 3AD

**Investigation**

**Summary of the current system based on interview from 8th Nov 2014**

The problems with the current system lie in finding out the optimal bus journey between bus stops. The tourist office in Cambridge uses printed timetables of buses provided by Stagecoach in Cambridge for the tourists to find how to get from place to place. This requires a lot of time as people have to look up a table. In most cases the bus journey involves at least one bus change. It is not easy to spot which bus stops share bus stops and where it is best to get on and/or off. The tourists often ask for help at the desk, unless the worker has memorised all the timetables the workers usually result to looking at the printed timetables themselves. This takes up a lot of time which could otherwise be spent answering other questions. The printed timetables are often given to tourists for their benefit but this is often unnecessary as only specific bus journeys are required. This system requires a lot of printing and is expensive. The printed timetables only show the name of the bus stops and not the address and although the bus stop name often corresponds to a nearby street name or facility, it is difficult to locate its actual location.

The printed timetables are purchased from by Stagecoach Cambridge for a reasonably small financial cost. The timetables are printed using the information from their database which gets updated at least once a year, so Tourist Information Centre has to request a reprint of their timetables to be kept up to date.

**Problems with the current system**

Here are the gathered problems with the current system:

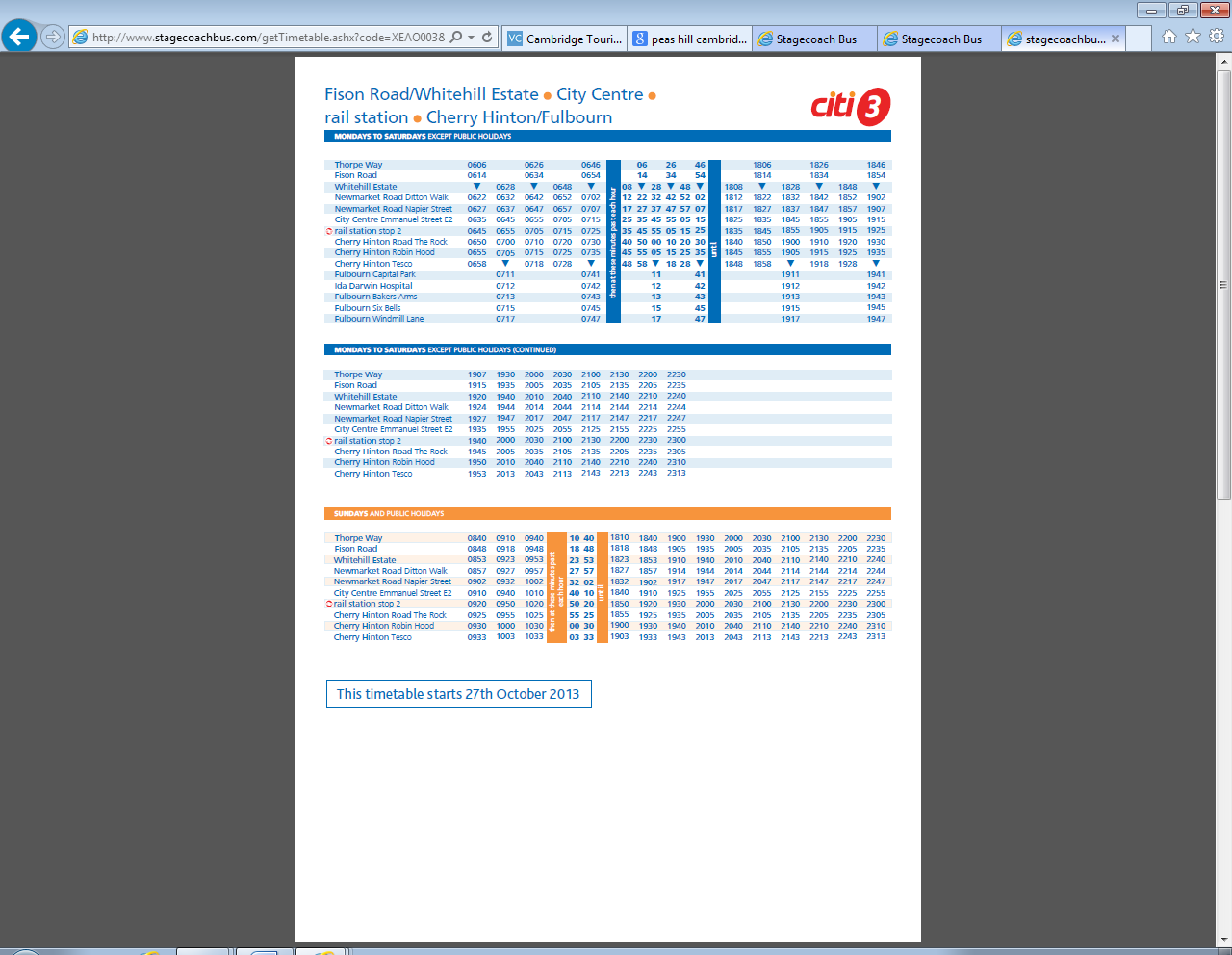
Printed timetables – needed for every bus coming from and to Cambridge, given to each tourist if they would like a copy of a timetable for themselves, costs money, uses a lot of paper and ink, hard to search through the timetable, hard to seek bus stops, time costly, only key bus stops are include in the timetable, reprinting required each time a timetable changes, only bus routes are given and one has to look through several bus routes to plan an appropriate bus journey, hard to find alternative bus journey.

Finding the appropriate bus stop – not all bus stops are included in the timetable, no data about the location of the bus stop apart from its name, no address, hard to locate on a map

Input, output, forms and report formats from existing system

**Documents of the current system**

**Example of a timetable for a bus used by the current system:**

****

**Data Flow Diagram of the current system**

Printed timetables stacks

P1

Bus journey

The worker from the tourist office/ the tourist looks at the printed timetables

1

Timetable Printing

The timetables are printed using data provided by Stagecoach

2

Destination and source bus stop

Request Timetables of buses

Timetables of buses

Timetables of buses in paper form

Request Timetables of buses

Timetables of buses

**Data in the proposed system**

**Data flow diagram**

A postcode/ address of a location to display nearest bus stops

Maps interface

Bus name

Destination and source bus stop

Find Bus stop

Find appropriate bus stop given a location

3

Find bus journey

The tourist office worked looks at the printed timetables

2

Bus route information

General information about the bus

1

Bus stop location

Bus info

Timetable info

Bus stop info

Request

Request

Request

D2

Database timetable and day category

Database bus stops

D3

Database bus routes

D1

Timetable and day category

Bus stops

Bus Routes

Update

Requesting an update for the database

2

A copy of the database bus information

Request

**Entity Relation Diagram**

Here is an entity diagram which shows the basic relationship between the tables in the proposed system. This is only a very basic design for a database to act like a Stagecoach database.

Timetable

BusRoutes

Busstops

DayCategory

**Data Volumes:**

**Background**

After doing some preliminary research, I found the following information about the Stagecoach bus network in Cambridge:

* There are 33 stagecoach buses coming to and from Cambridge
* Each bus has a 2 ‘bound’ route
* Each route has about 70 bus stops
* 6 buses are every 10 min, 10 buses are every 20 min and the rest 17 buses on average are every 45 min in a day
* The buses usually run from 6:00 until 21:00

**Data Dictionary and Data Volumes:**

Here is a basic design for the database to get an idea of how much storage would be required for the system.

**Bold** indicates a primary or in the case where there are several primary keys they make up a composite key.

Table name - Buses

|  |  |  |  |
| --- | --- | --- | --- |
| Columns | Type | Size | Number of bytes |
| **BusID** | Integer | Up to 66 | 1 |
| Name | String | 20 chars (ASCII) | 20 |
| Description | String | 200 chars (ASCII) | 200 |
| Number of bytes for each row | | | 221 |
| Total kB | | | 13.3 |

Bus Stops (there are about 2000 bus stops)

|  |  |  |  |
| --- | --- | --- | --- |
| Columns | Type | Size | Number of bytes |
| **Bus Stop ID** | Integer | Up to 2000 | 2 |
| Name | String | 80 chars (ASCII) | 80 |
| Latitude | Decimal | Between -90.0 and 90.0 | 4 |
| Longitude | Decimal | Between -180.0 and 180.0 | 4 |
| Number of bits for each row | | | 90 |
| Total kB | | | 1800 |

Day Categories (Mondays to Saturdays, and Sundays and Bank Holidays)

|  |  |  |  |
| --- | --- | --- | --- |
| Columns | Type | Size | Number of bytes |
| **Day Category ID** | Integer | Up to 9 | 1 |
| Description | String | 30 chars (ASCII) | 30 |
| Number of bits for each row | | | 31 |
| Total kB | | | 0.27 |

Timetable (70 bus stops on average for each bus route, 66 routes, on average 60 bus courses every day for each bus route, 3 day categories, and therefore about 830000 rows of data altogether)

|  |  |  |  |
| --- | --- | --- | --- |
| Columns | Type | Size | Number of bytes |
| **Bus ID** | Integer | Up to 66 | 1 |
| **Bus Course ID** | Integer | Up to 99 | 1 |
| **Bus Stop ID** | Integer | Up to 2000 | 2 |
| Time | Integer | 0000 to 2400 | 2 |
| Day Category | Integer | Up to 9 | 1 |
| Number of bits for each row | | | 7 |
| Total kB | | | 5800 |

All together: about 7600kB, which means that the size of the database would be about 7.6Mbytes.

**Constraints**

**Software and Hardware constraints**

The devices must have access to the internet.

**Time constraints**

The final outcome should be completed and ready by 27thFebruary.

**Users and Clients**

The users are going to tourists that not necessarily understand English the software should be in simple English for foreign tourists to understand

**Benefits of Computerising**

Allowing bus journeys to be calculated quickly and efficiently

Allowing bus stops to be searched by location

Allowing the tourist office to keep the timetables up to date

Allowing a more user-friendly interface

Allowing an extension to their already existing software and website

Allowing a user to print only what they need and saving paper

**Not included in the computerisation**

This solution would require software to manage the database and sync it with the Stagecoach database.

**Objectives of the new system:**

**Specific Objectives:**

1. The program should accept the start and destination bus stop, the desired time of departure and date.
2. Upon entering an address of a location, the program should output a list of nearest bus stops to that location on a map.
3. The program should show the shortest bus journey depended on the time and day of the journey.
4. The system should be compatible of handling about 7.6Mbytes of data at a time and the bus journeys should be displayed within 2 seconds.
5. At least three bus journeys should be displayed showing the time of start of the journey and the time of arrival at the destination, and number of bus changes, total time of journey.
6. Upon selecting a route more details should be shown including: the time and place of the bus changes, the bus name of the buses, time of the journey on each bus and waiting time between bus changes and the total waiting time.
7. The program should produce a document that could be printed or stored if desired. The information should include: the time and place of the bus changes, the bus name of the buses, time of the journey on each bus and waiting time.
8. When selected, the route will be displayed on a map, showing all bus stops and bus changes.
9. The map should have basic zoom in and out feature and span.

**General Objectives:**

1. Easy to follow and learn to use within a few seconds of working with the program
   1. The office staff should know how to use the application so that they can be asked for help
   2. The system should be simple enough for the non-experienced tourists to learn how to use it quickly
2. Easy to update information about timetables

**Extension Objectives:**

1. When clicked on a bus stop on the map the following information could be displayed: the buses going through the bus stop, geographical location.

**Possible solutions**

**Storage Solutions:**

Stagecoach does not provide access to their database but is willing to send copies of their timetables in electronic file, which can be easily pulled into a database or text file.

***Text File***- This solution of storing information would have a high to seek time as it would involve looking at about 200000 records

***Local Database*** - a database has existing optimised search algorithms, good for a desktop solution

***Server Based Database*** - hosting required – requires money, internet access – not too much of a problem. This type of solution would

**Program Solutions:**

***A desktop application***- has to be install on every computer, can only be accessed from the tourist office, could use both a local database or from a server, can be operating system specific in certain programming languages, slightly easier to develop

***A phone application*** - nearly everyone if not all have a phone, but operating system specific android, OS and Windows Phone, assumes every tourist has a suitable phone, could use a database but limited processing speed and storage space, a server based database may be required, with this solution printing could be avoided because a phones is portable

*My limitations*: I would need to discover a lot more about making phone apps, I would develop it the apps in C# which would only work on Windows Phones

***A web-based app and a server based database***- can be accessed from anywhere with internet connection, anyone would be able to access the application, cross platform, requires a server based database, accessible from both phone and desktop, calculation would be done on a server.

*My limitations*: I would need to discover a lot more about making web apps. There are different ways of creating web solutions; the most common one is MVC. I would require me to break through this type of style of programming.

**Chosen Solution:**

Desktop application - I choose this solution because I feel most confident in developing programs in this form. Windows Application Forms use XMAL which is a mark-up language so it will be capable of creating a friendly graphical interface.