An Investigation into Cross Platform Traffic Management Mobile Applications.

Niall Heffernan
B.A.I. Engineering
Final Year Project April 2013
Supervisor: Dr. Hitesh Tewari

School of Computer Science and Statistics
O’Reilly Institute, Trinity College, Dublin 2, Ireland
Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

_________________________  03 - April - 2013
Name
Permission To Lend

I agree that the library and other agents of the college may lend or copy this thesis upon request.

_________________________  03 - April - 2013

Name
Acknowledgements

I would like to thank my supervisor, Hitesh Tewari, for his support and guidance throughout the project, without which I would not have been able to complete it during my times of wavering motivation.

I would like to thank my family for their continued support all through the project, and indeed their support over the last 4 years.

I would also like to thank my classmates for me bouncing semi-random ideas and thoughts off them, for putting up with it and also for all those late night sessions coding and writing this report.

Finally I would like to thank everyone whom helped with this project by listening to me, by suggesting ideas, and by giving feedback on the idea, it wouldn’t be complete without you!
Abstract

The purpose of this thesis was to build a cross platform mobile application that could run in the background on a user’s global positioning system (GPS) enabled phone to automatically send reports on traffic congestion. There was also to be a server side implementation where approved users or organisations can view the current traffic congestion reports and further report them. The result was the creation of a cross platform application that realises these goals and is ready for public release. The server side was also created successfully and is available at http://www.trafficapplication.com.

The goal of the thesis is that in the future, a system such as the one designed could help drivers to avoid road traffic congestion and enhance the quality of traffic reporting.
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Dublin Traffic Cameras</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Garda Twitter</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>AA RoadWatch</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Mobile Millenium</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>Google Maps - London</td>
<td>16</td>
</tr>
<tr>
<td>3.1</td>
<td>Smart Phone Operating Systems</td>
<td>17</td>
</tr>
<tr>
<td>3.2</td>
<td>PhoneGap Wrapper</td>
<td>20</td>
</tr>
<tr>
<td>3.3</td>
<td>PhoneGap Architecture</td>
<td>21</td>
</tr>
<tr>
<td>3.4</td>
<td>Demonstrator Application</td>
<td>25</td>
</tr>
<tr>
<td>3.5</td>
<td>Application States</td>
<td>26</td>
</tr>
<tr>
<td>3.6</td>
<td>Standard Resource Usage</td>
<td>29</td>
</tr>
<tr>
<td>3.7</td>
<td>Demonstrator Resource Usage</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Client - Server Model</td>
<td>31</td>
</tr>
<tr>
<td>4.2</td>
<td>Website Sitemap</td>
<td>36</td>
</tr>
<tr>
<td>4.3</td>
<td>Database Model</td>
<td>37</td>
</tr>
<tr>
<td>4.4</td>
<td>Application Designs</td>
<td>40</td>
</tr>
<tr>
<td>4.5</td>
<td>Man in Middle Attack</td>
<td>44</td>
</tr>
<tr>
<td>5.1</td>
<td>Splash Screen</td>
<td>49</td>
</tr>
<tr>
<td>5.2</td>
<td>Information Screen</td>
<td>49</td>
</tr>
<tr>
<td>5.3</td>
<td>Notification Sent Confirmation</td>
<td>50</td>
</tr>
<tr>
<td>5.4</td>
<td>Timeout Notification</td>
<td>51</td>
</tr>
<tr>
<td>5.5</td>
<td>Last 5 Alerts</td>
<td>51</td>
</tr>
<tr>
<td>5.6</td>
<td>Parking Information</td>
<td>52</td>
</tr>
<tr>
<td>5.7</td>
<td>Road Information</td>
<td>53</td>
</tr>
<tr>
<td>5.8</td>
<td>Website Home</td>
<td>54</td>
</tr>
<tr>
<td>5.9</td>
<td>404 Error</td>
<td>55</td>
</tr>
<tr>
<td>5.10</td>
<td>Password Notification</td>
<td>55</td>
</tr>
<tr>
<td>5.11</td>
<td>Activation Email</td>
<td>55</td>
</tr>
<tr>
<td>5.12</td>
<td>Incident Table</td>
<td>56</td>
</tr>
<tr>
<td>FIGURE</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.13</td>
<td>Incident Map</td>
<td>57</td>
</tr>
<tr>
<td>5.14</td>
<td>User Details</td>
<td>58</td>
</tr>
<tr>
<td>6.1</td>
<td>Screenshot - iOS</td>
<td>64</td>
</tr>
<tr>
<td>6.2</td>
<td>Screenshot - Android</td>
<td>64</td>
</tr>
<tr>
<td>6.3</td>
<td>Screenshot - Windows Phone 7</td>
<td>65</td>
</tr>
<tr>
<td>6.4</td>
<td>Windows Phone Emulator Issue</td>
<td>67</td>
</tr>
<tr>
<td>6.5</td>
<td>Windows Phone View</td>
<td>68</td>
</tr>
<tr>
<td>6.6</td>
<td>Xcode Locations</td>
<td>69</td>
</tr>
<tr>
<td>6.7</td>
<td>Application Resource Usage</td>
<td>70</td>
</tr>
</tbody>
</table>
Contents

Declaration ii
Permission To Lend iii
Acknowledgements iv
Abstract v

List of Figures vi

1 Introduction 1
  1.1 Objectives . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  1.2 Aims . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
  1.3 Ideal Solution . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
  1.4 Thesis Organisation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2 Traffic Management Approaches 4
  2.1 Current Systems . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
    2.1.1 Fixed Systems . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
    2.1.2 Mobile Systems . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
    2.1.3 Others . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
  2.2 Traffic Reporting . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
    2.2.1 AA RoadWatch . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
    2.2.2 Radio . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
    2.2.3 Traffic Message Channel . . . . . . . . . . . . . . . . . . . . . . . 11
    2.2.4 Other Methods . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
  2.3 Similar Projects and Applications . . . . . . . . . . . . . . . . . . . . . . 12
    2.3.1 Mobile Millenium . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
    2.3.2 Other Mobile Applications . . . . . . . . . . . . . . . . . . . . . . 14
    2.3.3 Similar Websites . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

3 PhoneGap 17
  3.1 About PhoneGap . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
### CONTENTS

3.1.1 Technology ........................................ 19
3.2 PhoneGap Applications Versus Native Applications ................. 20
3.3 Other Cross Platform Frameworks ............................ 22
  3.3.1 Marmalade SDK .................................. 22
  3.3.2 Feed Henry ..................................... 23
  3.3.3 QT Framework .................................. 23
  3.3.4 RhoMobile ..................................... 24
  3.3.5 Appcelerator .................................... 24
3.4 Demonstrator Application .................................. 24
  3.4.1 The Application .................................. 24
    3.4.1.1 Functionality ................................. 25
  3.4.2 Accomplishments ................................ 27
  3.4.3 Resource Usage ................................ 28

4 Design ................................................................ 31
  4.1 Technologies Used ...................................... 31
  4.2 Design Of System ....................................... 32
    4.2.1 Client Side ..................................... 33
    4.2.2 Server Side ..................................... 35
      4.2.2.1 Website .................................... 35
      4.2.2.2 Database .................................. 37
  4.3 Languages And Frameworks ................................ 38
  4.4 Technical Approach ..................................... 39
    4.4.1 User Interface ................................... 39
    4.4.2 Background Operation ............................ 40
    4.4.3 Algorithm ....................................... 41
  4.5 Security Considerations ................................ 43
    4.5.1 Attacks .......................................... 43
    4.5.2 Incorrect Usage ................................. 45
    4.5.3 Other ............................................. 45

5 Implementation .................................................. 47
  5.1 Client Side ............................................... 47
    5.1.1 Features ......................................... 48
  5.2 Server Side ............................................... 53
    5.2.1 Website .......................................... 54
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.2</td>
<td>Database</td>
<td>57</td>
</tr>
<tr>
<td>5.3</td>
<td>Implementation Issues</td>
<td>59</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Permissions Issue</td>
<td>59</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Technological Issues</td>
<td>60</td>
</tr>
</tbody>
</table>

## 6 Evaluation

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Testing Method</td>
<td>61</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Client Side</td>
<td>62</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Server Side</td>
<td>65</td>
</tr>
<tr>
<td>6.2</td>
<td>Testing Issues</td>
<td>67</td>
</tr>
<tr>
<td>6.3</td>
<td>Testing Results</td>
<td>69</td>
</tr>
</tbody>
</table>

## 7 Conclusions

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Future Work</td>
<td>73</td>
</tr>
</tbody>
</table>

## Electronic Resources

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Application Information Page</td>
<td>80</td>
</tr>
<tr>
<td>B</td>
<td>Change Log</td>
<td>82</td>
</tr>
</tbody>
</table>
1 Introduction

The problem of traffic congestion and how to manage it is becoming an ever-increasing problem for authorities. The lack of real time traffic information in many instances hampers both authorities and reporters in attempting to assist drivers to navigate around congestion.

Road traffic congestion has become a serious issue for both authorities and drivers, it costs thousands of hours a year in productivity, adds to driver frustration and is detrimental for the environment.

A report entitled The TomTom Congestion Index Q3 2012, listed Dublin, Ireland as having an average of 26% congestion, spiking to 58% during the evening rush hour commute [1]. This report also showed how congestion has increased significantly in the last 5 years. Further, it is estimated that traffic congestion costs Dublin businesses €2.5 Billion each year [2].

1.1 Objectives

The objectives of this project were to create a mobile application that has the ability to run across multiple mobile operating systems on devices that have GPS support. The application should have the ability to automatically monitor traffic conditions and detect when and where congestion occurs.

The application should also have a limited number of user controls and inputs to signal other events such as traffic light faults and road traffic collisions.

There is also a requirement to have a server side implementation for the project, so that approved organisations can view the incidents occurring on a real time basis. The organisation must be pre approved by an administrator so as to ensure data privacy is maintained at all times.
1.2 Aims

There are multiple aims for this project, to investigate whether it is possible to develop a cross platform mobile application that can automatically detect and report on traffic conditions based on location features is however the main. These reports will then be viewable by approved parties on a server side implementation, preferably a website with a mapping feature.

The application is to be a cross platform application. Cross platform applications have the ability to be made once and run on the different operating systems available, such as iOS, Android, Windows Phone and Blackberry.

Another aim of the project is to check if it is possible to automatically detect traffic conditions based on the geo-location features, in this instance the speed of the vehicles is used.

A server side implementation to be able to catch the reports and view them on a mapping system is also a required aim, as without such a service to view the incidents, the client side system would be under-utilised.

1.3 Ideal Solution

The ideal solution to this problem of identifying congestion is that each car would have traffic sensors available in built that can monitor traffic in the area and report on conditions to a central database. This would also ensure that the central database is receiving enough reports to ensure data integrity.

The approach taken upon in this project is somewhat similar, the sensors however are only GPS enabled mobile smart phones that have downloaded this application and are sending updates. To that effect it is possible in many instances that not enough reports will be received, or that spurious reports could be received.

To ensure data integrity the reporting agency, in an ideal world, would also have access to traffic cameras in the area and other resources to be able to manually check and confirm any reports received.

This ideal solution is however at the current time unrealistic due to the amount of hardware that would be required to equip and maintain vehicles with the sensors. Many traffic cameras already exist currently throughout Dublin city and are provided by Dublin
1.4 Thesis Organisation

This thesis is organised into a number of chapters of specific areas and is further subdivided into sections to ensure it is easy to navigate and understand.

Chapter 1, this chapter contains a basic introduction, introducing the reader to the project and its aims.

Chapter 2 looks in depth at the current systems for monitoring traffic, how traffic congestion is reported currently in different areas, both in Ireland and across the world. This chapter also examines what similar work has been done in both academic projects and commercial projects.

Chapter 3 looks at the cross platform framework tool utilised, which is called PhoneGap. This chapter looks at the advantages and disadvantages of programming cross platform applications in such a manner. It also examines the demonstrator application created that was designed to test the PhoneGap APIs and geo-location features.

Chapter 4 is a chapter dedicated to the design of the application, both on the client side and on the server side. This chapter describes the design, the approach, and the security considerations.

Chapter 5 details a full explanation of the implementation, including any issues which arose during implementation.

Chapter 6 details the testing and evaluation strategy, as well as the results.

Chapter 7 features a conclusion to the project and examines where future work could be carried out if required in the area.

Chapter 8 is the bibliography listing any references and citations used throughout the project.

There are two appendices included, the first is the information which is displayed to the application user on first install. The second is a list of any application changes committed to the version control during development.
2 Traffic Management Approaches

Traffic congestion is a condition that occurs as road use increases, it is characterised by lower speeds, longer trip times and increased vehicle flow [4]. A traffic jam is a situation when vehicles are stopped for a large period of time.

Some cities try to target this problem with features such as one-way systems, and features such as congestion charges, such as those implemented in London [5]. Each of these however is not a solution, indeed nothing by itself will be a successful solution, and rather each of these exercises exists to try to limit the effect on a short-term basis of traffic congestion.

By implementing better quality reporting and helping drivers to avoid traffic congestion is the most ideal solution and way forward given current infrastructure. The majority of roads are never congested, and all congestion reports and indexes simply examine the congestion of major routes. By alerting drivers to the location of congestion and road traffic incidents, it is possible to route away from the problem. This is all made possible by quality reporting of the incidents and delays, which could potentially be aided hugely by automation of the process.

In Dublin, between 2005 and 2010 there was a 57% increase in the number of cars designated as taxis during peak periods on the roads [6]. Similarly, there was an overall increase of 3.7% in the number of cars. This all has an impact on the amount of traffic congestion experienced in the city. Many of the worlds roads were built for much less traffic, and this is true for Dublin, where many roads date back hundreds of years. In the USA, traffic jumped 236% for a 20% population rise between 1982 and 2001, this trend looks set to continue with the number of cars in the world set to double between now and 2020 [7]. The number of cars world wide in 2010 was thought to be 1.015 Billion [8]. With increases such as these being experienced, congestion is an ever increasing problem.

Traffic congestion has a number of negative impacts; there is the environmental impact from increased air pollution and greater usage of natural resources such as petrol and diesel, there is the encouragement of road rage incidents as drivers are held up in traffic for multiple hours. There is a significant wear and tear factor on vehicles as a direct result of idling in traffic and the frequent acceleration and braking that this causes. However by far the greatest detrimental effect is the loss of productivity of staff and the hours lost
in traffic, which has a huge economical cost [2].

There are 6 classifications of traffic congestion, as defined in the Highway Capacity Manual, published by the Transportation Research Board of America [9]. These six levels, from A-F, are called the level of service, with A being free flow and F describing a total breakdown in vehicle flow. These level of service indicators are used worldwide to help establish procedures and manage traffic on a reactive basis.

## 2.1 Current Systems

There are currently many various ways to monitor and control traffic operations. There are various types of fixed and mobile services and each depends on the level of service before being activated, however some methods are permanently activated features.

The main route to activation for many of these features depends on them being reported correctly, there are also a number of reporting avenues available for a motorist to both reports problems and to receive reports on problems.

The tolerated level of congestion is somewhat linked to the amount of money the society is prepared to pay for traffic management systems [10].

Many of the systems currently in place are however delayed in their implementation and not real time.

### 2.1.1 Fixed Systems

The majority of traffic management systems are fixed in their nature. Fixed traffic management systems are installed in known hot spots for congestion or across wide areas to monitor congestion on a rolling basis.

Many fixed systems are directly related to the road infrastructure available and need to be designed and implemented during construction of the roads. This system is adequate for new roads being created but will not help to solve any congestion problem that is currently occurring on roads. Fixed systems that can be implemented during construction include bus lanes, high occupancy lanes and vehicle grade separation.

The most commonly employed fixed method of monitoring and controlling traffic conditions is the use of traffic cameras. Cameras are set up to monitor traffic from an elevation in strategic areas. For example in the Dublin area, Dublin City Council have
2.1. Current Systems

a range of roughly 180 cameras (Figure 2.1) that are monitored in the Dublin Traffic Control Centre [3]. These cameras provide operators with a first hand look at traffic conditions throughout the city and give them the ability to react quickly to any traffic that may be building.

![Figure 2.1: Some of the traffic cameras that cover Dublin City.](image)

Traffic sensors are an ever-increasing solution being used, specifically at night-time and times of lower traffic levels. These involve setting sensors either under the road or as cameras, to monitor the presence of vehicles at junctions. The major road is given priority throughout unless a vehicle is detected at the minor road; in this situation the minor road is given brief priority. This system, when enacted is significantly better than fixed systems which require motorists to wait when there is no other traffic on the minor road.

Variable speed limits are systems that are mainly applied to high-speed roads and are used to increase the capacity of a road. Vehicles at higher speed cannot move in close proximity to each other, however by decreasing the overall speed, vehicles can travel closer to each other thus increasing the capacity of the road. This method is used extensively in the UK [11].

Dublin City Council has introduced a system to help with traffic management in Dublin city centre by designating one area a bus only area in rush hours. This has the effect of making transport for buses faster, but the traffic intensity on surrounding streets grows during this time. It is used as an incentive for public transport. Similar to this in other cities is automatic barriers to control access to certain streets and areas during
2.1. Current Systems

peak periods.

Fixed speed sensors are sensors that are in place on the roadway. They monitor the speed of passing vehicles. Using the speed it is simple to calculate whether congestion is being experienced or if the section of road is in free flow. These are extensively used in Dublin in areas that are not covered by cameras.

One-way systems are an approach that have been used across the world for many years. In some cities one-way systems are however only activated at certain times when traffic conditions dictate, as defined by the level of service categorisation.

2.1.2 Mobile Systems

Fixed systems are of specific use, but are however limited to the geographical spread of where they can be implemented and the times in which they can be implemented.

The most popular method of traffic control in a mobile environment is through the use of reporting to help alleviate traffic. Improving the quality of reporting is one of the aims that this project wishes to accomplish. By improving the quality of reporting the motorists have a significant reason to avoid heavily trafficked areas.

Mobile message boards are a tactic employed during known busy times, especially in the case of special events occurring in the area. These message boards can warn motorists to avoid certain areas, which will help alleviate conditions in the particular area.

2.1.3 Others

Increasingly many motorists are taking to social media to report problems with vehicle flow and to check for problems. Social media such as Twitter has been the subject of many recent studies looking at the possibility of harvesting data about traffic conditions for a certain area [12]. Through this authorities can also issue alerts to motorists to help limit traffic impact. Figure 2.2 shows an example of such.

The simple act of changing school opening hours by as little as half an hour could have a huge effect in some areas on traffic. This has been tested and implemented in many different countries. By altering the opening times so that they do not collide directly with the rush hour commuting times takes thousands of vehicles off the road in a small area.

Many cities, such as London have introduced charges called congestion charges. These
2.2 Traffic Reporting

Traffic reporting is the near real time distribution of information about road traffic conditions to motorists. The reports help motorists to anticipate traffic congestion and potentially reroute [13].

are charges directly related to the management of traffic, it is essentially an extra charge on driving in a certain area, and has the effect of encouraging motorists to both car pool and use other means of transportation.

Large media organisations and police utilise helicopters to get a wide-angle view on the conditions of an area. This method is used widely in cities throughout the world, in some instances, such as AA RoadWatch, this has however been overtaken by traffic cameras.

One final method is the use of the local police in an area, who will know the road conditions and area very well. Police are often mobilised to divert some drivers away from specific routes and redirect motorists away from traffic congestion. This is used extensively in cities such as Dublin during the busy Christmas period but is to be avoided on a regular basis.

2.2 Traffic Reporting

Traffic reporting is the near real time distribution of information about road traffic conditions to motorists. The reports help motorists to anticipate traffic congestion and potentially reroute [13].
2.2. Traffic Reporting

There are many ways to report traffic conditions, and for motorists to be informed of the conditions. Reporting on radio initially began in the 1970s in America as a result of the increasing traffic volume. During the 1980s and 1990s the competition began to grow. During this time the quality increased also. In recent years, technical innovations have combined to make more reliable and frequent reports, with online reporting now a factor and various crowd-sourced methods of collecting data also.

2.2.1 AA RoadWatch

AA RoadWatch is a division of the Automobile Association of Ireland that reports on traffic conditions throughout Ireland. They provide an online website to view all incidents currently taking place (Figure 2.3) and also provide regular radio bulletins. The radio bulletins are broadcast on partner radio stations who use AA RoadWatch for all their traffic reporting, tailored to the particular broadcast area [14].

AA RoadWatch has a number of methods it uses to gather information. The first is a primarily research based role, they provide alerts on long term roadworks and on special events that may cause traffic disruption, this knowledge is received from building contractors and event promoters, eager to lessen the burden of their activities, but also through other research of media and events to find a list of scheduled events.

AA previously had a helicopter covering the Dublin area and any major events between 1995 and 1999 but this was later retired as other information sources became prevalent, particularly the advent of traffic cameras throughout Dublin.

As mentioned earlier, social media is becoming an increasingly popular form of reporting, and AA RoadWatch utilise the service heavily. It has an "extremely successful method, we get lots of useful information from motorists on the move through this medium", Aoife Carragher, AA RoadWatch Co-Ordinator November 2012.

Dublin City Council provides the bulk of incident reports, providing information via their live control centre for the Dublin area. In turn they receive their reports through their operators who constantly monitor the traffic conditions via the traffic cameras, and through their Live Drive radio service, which is discussed in the next section.

A major source, and by far the most inefficient is to gather information by ringing around to local Garda stations. During some days, AA RoadWatch staff ring stations multiple times to either enquire about incidents or ask about current conditions. This is obviously highly inefficient and in need of change, in this context however, the information
would be highly accurate in general.

The AA have extremely good relationships with many business and bus companies, who report problems on the road network, the National Roads Authority are a source for any longer term problems and the Motorway Control Centre provide information on the motorways. All of these sources are however inconsistent in their reporting, as there is often little incentive for them.

Increasingly, AA RoadWatch are also accessing satellite information about traffic flows, which is an area of huge growth and research. This area involves a satellite being set up to compare previous pictures versus current pictures to attempt to determine traffic flow.

### 2.2.2 Radio

Radio is the means by which the vast majority of incident reports reach a motorist who is travelling, and is therefore the critical medium. It would be envisaged that the application reports from this project would be made available to not only organisations such as AA RoadWatch but also the interested radio stations.
2.2. Traffic Reporting

There are a huge number of radio stations in Ireland, some broadcasting nationally and some only broadcasting to specific areas or regions. Having contacted many of these stations to enquire about their reporting methods, the majority receive reports direct from AA RoadWatch or in fact broadcast the bulletins produced by AA. In addition they receive some other reports from their individual listeners.

One radio station worth mentioning is Dublin City FM, and a particular show on that station called Live Drive. This is a Dublin City Council funded volunteer radio station. Live Drive broadcasts twice a day, in the morning rush hours and in the evening rush hours (7am-10am and 4pm-7pm) directly from a studio situated inside the Dublin Traffic Management Centre. Gerry Cahill, one of the station directors, states that Live Drive also acts as a crucial method of Dublin traffic controllers sourcing information via listeners also.

The service provided by Live Drive is not only a service of providing information on the traffic conditions across the city but also providing listeners with alternative routes.

2.2.3 Traffic Message Channel

A feature that can be built into FM radio stations is a feature called Traffic Message Channel (TMC). TMC is a technology for delivering traffic information to motorists. In Ireland, TMC is active nationwide and is available on RTE Radio 1 [15].

Each known incident is binary encoded and sent as a TMC message. Each message consists of an event code, location code, expected duration and the affected extent. Location codes are maintained on a national level.

TMC is a low bandwidth system, which is received by a TMC receiver. The receiver can decode and display the messages. The decoded messages can be shown visually or be played back over audio.

There are two main problems with TMC. The first is Italian researchers have proved a security issue in relation to the sending of the packets and possible compromise, however the use of an attack is very limited. The second issue relates to visual displays of information and the potential distraction for the driver. This in many countries is a legal issue and a safety issue, this issue has caused automobile companies to be slow to role out the service on a widespread basis.
2.2.4 Other Methods

There are many new reporting systems coming into place in recent years. The majority of these are online based and in real time.

In recent times Dublin City Council has made still images from its traffic cameras available online, allowing the public to personally check many of them for traffic conditions in an area or along a route before setting off.

As has been mentioned above, incidents are now often publicised on social media. Services such as Twitter allow users to view incidents from anywhere and receive up to date information about traffic conditions.

2.3 Similar Projects and Applications

The area of traffic management and traffic reporting using mobile phones has received considerable interest lately and many products have been released in the area. A large amount of research has been carried out also. [16–19]

Much of this research and work is in the same area as this project, however there is no identical system known. The majority of work is based on the Mobile Millennium project, as described below. Projects such as these report on traffic conditions but are programmed to only work within a small geographic area of known roads, or in a small area where the congestion characteristics are well known. This project aims to detect congestion with no prior knowledge of the area that it is testing.

2.3.1 Mobile Millennium

The Mobile Millennium project was a research project that used GPS in cellular phones to gather traffic information, process it and distribute it back to phones. The research was carried out by the California Centre for Innovative Transportation (CCIT) in UC Berkeley, San Francisco, California [20].

The mobile millennium project integrated numerous feeds into traffic models, which were generated and processed in real time, with the results fed back to the devices. The feeds included data obtained from GPS enabled phones, all of San Franciscos taxis through GPS, traffic loop detectors and historical data.

Mobile Millennium provided a consumer tool more so than a traffic management tool,
2.3. Similar Projects and Applications

with the data being fed back to the user on their device for the user to visually see which
routes are congested based on colours along road sections. There were roughly 2000
volunteers used in testing of the product.

![Image](image_url)

Figure 2.4: Results of Mobile Millennium project for the San Francisco area.

There are however a number of limitations for the project. The project would only
work in a small area where the algorithm was aware of the road being travelled, its speed
limit, the location of junctions and the location of traffic lights. The result of this is that
the project is not easily scalable or transferable to a different location.

The project in itself was a continuation of the Mobile Century project, an even more
limited project whereby vehicles drove a set 10-mile stretch of road during certain times
and the congestion was calculated based on resulting data.

The millennium project was expanded beyond this, but still used the theory of virtual
trip lines, which requires heavy data processing and knowledge of the operation area.

The main disadvantages of the project apart from the lack of scalability are issues on
the client end. The location data is constantly sent to the server and evaluated along
with all the other data being received, and then the results of the processing are returned.
This has a huge effect on the phones battery life and on the network usage of the phone,
which would be considerably higher. In addition it has to be a conscious decision by the
operator to turn the application on, as the application is required to run in the foreground
of the device.
2.3. Similar Projects and Applications

The difference with this project is that the application is passive in nature, running in the background using minimal battery and network resources.

The mobile century and mobile millennium projects are still highly significant however as they helped to overcome many main problems and set the groundwork for many similar projects soon after. The data from the mobile century project has been released recently, which shows all the data collected and analysed as part of the project, while the same latitude, longitude and speed attributes are analysed similar to this project, the implementation is significantly different.

2.3.2 Other Mobile Applications

There are many client applications available which, similar to the Mobile Millennium service, provide a user service for a fixed area rather than a traffic management service.

One such service is an application called Waze [21]. Waze is a community based mapping service that has inbuilt navigation and real time traffic, the maps are submitted by users, with traffic information embedded also. Waze provides a navigation service whereby users can find directions to their final location based on the route with the least traffic congestion. Similar to the Mobile Millennium project this application is only suitable in specified mapped areas, and the navigation quality is dependent on the community maps of the area. In addition, the application is also very bandwidth intensive as it relies on constant updates being sent and received.

An Irish based application; iTraffic [22] is a GPS navigation system with real time traffic updates. iTraffic works in a unique way, and is a user-based product also. There is a method called Cellular Floating Vehicle Data (CFVD) upon which this application is based. Within a mobile network a data record is generated every time a phone hands over between cell towers. By matching these cell tower paths and the time between them, it is possible to calculate speeds and deduce traffic information. iTraffic also uses other sources of information such as historical data and known events in its estimations. The main disadvantage once again is the bandwidth use and small geographical scope of this application, as well as the known inaccuracy related to CFVD [23]. CFVD will not specify particular incidents, rather simply relate to congestion on a stretch of roadway, with the stretch dependent on the cellular towers in the area.

Simpler applications such as SigAlert are based on speed, but are not real time, as they do not receive data from the user base. SigAlert uses speed data of freeways based
2.3. Similar Projects and Applications

Applications currently available do not perform a traffic management service in the same way that this application sets out to, applications available are more of a service aimed towards the user, and the majority, if they utilise data from the phone GPS require large amounts of data, battery and bandwidth. Additionally each of the above operate of stretches of roadway as opposed to specific incidents.

2.3.3 Similar Websites

Multiple websites have been created that perform many of the same functions as the applications above.

The first example is one of the pioneers of real time traffic information, Google is the largest website in the world with an extremely comprehensive mapping service available. Members of the public who use Google maps for mobile and whom have GPS enabled on their phone, help Google to crowd source valuable information on speed and location. Speed and location updates are transmitted to Google servers where they are combined with information from other users in the locality; traffic layers can then be calculated with the appropriate colours overlaid on the mapping service to show the level of congestion [25].

The Google system was one of the first real time traffic information systems. The disadvantages of the system are the large amounts of data transmitted and the resulting large amounts of computations carried out on the server. Additionally the system will only work for specified routes, which are constantly being expanded by Google to achieve widespread coverage.

FreeFlowApp is a web application developed for the Dublin area by James Eggers, winner of the BT Young Scientist award; the application estimates traffic congestion in areas covered by Dublin road traffic cameras and provides incident information for incidents across Dublin via Twitter. The application examines the traffic camera pictures to analyse the concentration of vehicles in them, providing an estimate on the conditions based on the results. Incidents are found via twitter searches looking for location data and words such as accident and crash. This information is also fed to the application to attempt to provide information on current incidents.

FreeFlowApp is an innovative method, but is problematic as well, as it depends on
the traffic cameras being updated, the accuracy of the algorithm and it also has a limited range, as there are a limited number of cameras with a limited field of vision. The twitter search for incidents is also very innovative but relies heavily on Twitter users to accurately talk about incidents, using the correct words. The system is also very open to false results of a users location being reported wrong on Twitter or referencing incidents in different locations.

Figure 2.5: Google Maps with traffic overlay for London area.
3 PhoneGap

The PhoneGap mobile development framework was picked to be utilised for this project [26]. There are a number of cross platform mobile development frameworks available, each with advantages and disadvantages as outlined later in this chapter.

PhoneGap provides support to a wide number of applications and operating systems, is free, and has a wide range of functionality.

The need to build applications in a cross platform manner is unquestionable. Over the past 5 years, the evolution of the smart phone has been huge. During this period an ever increasing amount of mobile operating systems has also been developed. The mobile operating system market is currently extremely fragmented, as Figure 3.1 shows. Cross platform application development allows the developer to reach the majority of this market.

![Figure 3.1: Showing the distribution of smart phone operating systems.](image)

3.1 About PhoneGap

PhoneGap is a mobile development framework; it enables programmers to build applications for mobile devices using web technologies such as JavaScript, HTML and CSS.
The resulting applications are then not truly native nor truly web based, relying on multiple API calls to execute successfully.

PhoneGap was first developed at a development camp in San Francisco in 2009. It is used as the background framework for many other mobile application platforms. Adobe Inc. purchased the original developer (Nitobi software) in 2011, following this, the code was contributed to the Apache Software Foundation, a non-profit corporation to support Apache software projects.

The underlying software of PhoneGap is Apache Cordova, an open source set of device APIs that allow a developer to access native device functions. The apache Cordova project is regularly updated with new API calls and features. PhoneGap is a distribution of Apache Cordova, however is based purely on the project and contains no additional features at present, only a simple name change. PhoneGap is the only current distribution of the Cordova project [27].

The programming of a PhoneGap based mobile application is carried out in the applications specific integrated development environment (IDE), meaning that an iOS (Apple mobile operating system) application is created in Xcode, an Android application is created in Eclipse and a Windows Phone Application is created in Visual Studio, PhoneGap can be thought of as a wrapper to the web code that is produced, ensuring that the application runs on the native device correctly.

Perhaps the most powerful feature of PhoneGap is the large range of support available for it, because of the open source nature many of the developers have also written some excellent documentation on the project, and there is a range of discussion boards and forums for PhoneGap issues, many moderated by the developers and long term users of the project. There are also regular community events and web conferences held to expand a developers knowledge of the product.

Within PhoneGap, there is a wide range of support for so-called plug-ins, a plug in is a software component that adds specific ability to a larger software application. In the PhoneGap context, the plug-ins have the ability to add extra API calls for specific needs, an example of this is an implementation of a barcode scanner, or a PayPal plugin. These plugins can however be problematic if the developer were to install them without full knowledge of their workings.

Porting the PhoneGap code to another implementation of a mobile application is very simple. All of the PhoneGap code is created and modified in one central folder, called the "WWW" folder. From here, all the API calls are completed. To move a project from
3.1. About PhoneGap

an iOS implementation to an Android implementation is a simple task of copying this folder from Xcode and pasting into Eclipse. The only required change is changing the PhoneGap file to ensure the correct device API’s are utilised. This is an extremely simple procedure and an advantageous aspect of PhoneGap.

A service under the PhoneGap name, powered by Adobe, called PhoneGap Build is a new service designed to compile the applications in the cloud rather than on the users computer. This adds the functionality of letting a user compile and build applications for systems they previously had no access to, for example a Windows user can now create and compile iOS applications via the online PhoneGap Build service. This is a huge advantage to some developers and is provided by the Adobe corporation, completely separate to the PhoneGap project. It is free to use for open source applications and paid accounts are available for private applications.

Applications currently using PhoneGap and Cordova are widespread, from high profile applications such as the BBC Olympics application to lesser-known uses. As of PhoneGap version 0.8, PhoneGap applications are approved by the Apple App Store, subject to the standard terms and conditions of the store and standard testing. Previous to version 0.8 (October 2009) many PhoneGap and Cordova based applications had been rejected by Apple due to their nature.

Current supported operating systems are iOS, Android, Blackberry OS, WebOS, Windows Phone 7, Windows Phone 8, Symbian, Bada.

3.1.1 Technology

As previously mentioned, PhoneGap allows a developer to build native applications for mobile devise, both smart phones and tablets using web technologies. A developer builds an application using the supplied APIs to access device functionality.

HTML and CSS are used to render the view, while the calculations and logic of the application are carried out by using JavaScript. When a developer implements a feature in an application, the application calls the API using the JavaScript, the PhoneGap script then translates the call into a native API for the feature on the device.

The application shown on screen to the user is essentially a one-page web view, when the user interacts with the application; JavaScript code is run to process the interaction. A web view is a native application component that is used to render web content within a native application window. The application when running is essentially like any other
3.2. PhoneGap Applications Versus Native Applications

A web application, and runs in the same manner, it is however wrapped by the PhoneGap code to allow it to run on the device and contains custom PhoneGap API calls. A typical mobile web browser would not have access to device side components.

Figure 3.2: How PhoneGap wrapper operates

As new versions of the framework are released, new API calls are added. There is a clear roadmap for releases including the addition of new API calls and the resolving of any occurring bugs. In addition the framework is constantly expanding to take into account new devices that are appearing on the market. The most recent at the time of publication of this report being the Windows Phone 8.

The PhoneGap approach of developing using a web view and web languages is extremely useful and provides a fast development environment for many developers already experienced in web design. Additionally many applications can be based on a mobile website version of sites and are particularly versatile.

3.2 PhoneGap Applications Versus Native Applications

There are numerous advantages and disadvantages to developing a cross platform application, specifically to developing a cross platform application with PhoneGap. Many of these are discussed below; the overall decision should depend on the type of application being developed and its intended use.

The main advantage of building a cross platform, hybrid application is that the development time is significantly reduced, in particular for web developers who are experienced using the web languages. One language set can be used on every available platform, with
3.2. PhoneGap Applications Versus Native Applications

the result that the developer does not have to learn multiple complicated languages to get the same reach on an application. This is also of specific interest to smaller companies who will now only have to hire one developer competent in web languages as opposed to having to hire multiple if native applications were required, thus reducing costs.

Since PhoneGap is open source and free, there is a huge community of developers and supporters available with some extremely good documentation covering a wide range of issues, while this is also the case for native applications, in many instances the developer does not have the same level of control and assistance available.

There are however some major disadvantages to hybrid applications, one of the major problems for many people is the loss of the native feel to the application, the native feel referring to the look of the user interface. While this in many cases can be changed extensively with heavy CSS and HTML editing, the application will never truly have a native feel.

The majority of a PhoneGap application is programmed within one file, rather than multiple classes, all the development work is carried out within one folder. This is of a huge advantage from a security point of view, as it ensures there is less places for memory leaks to occur, and less problems with the visibility of variables and functions between classes, in addition, the application can be protected further by introducing reasonably simple secure commands in the JavaScript code to stop unwanted scripts running.

The basic structure of a PhoneGap application is detailed below in the figure 3.3.

![PhoneGap Architecture](image)

Figure 3.3: Basic PhoneGap architecture.
3.3. Other Cross Platform Frameworks

There is a danger of spending a lot of time on bug testing. The application when built to be a hybrid cross platform application will be extremely quick to build. The operating systems are not designed to handle web languages and can at times struggle with some implementations and designs. These can mostly be fixed however with debugging. When the code is transferred from one operating system implementation to another, there is also small quirks and differences, each of which need to be tested and fixed.

A native application is an application that is designed for one specific operating system using the programming language specific to that operating system. Native applications are the standard approach and have existed for a lot longer than hybrid applications. The main advantage of a native application is the speed that the application can run at. It is not advised to run a hybrid application for any function that will require too much CPU or graphics processing. For example, a game application should not be developed in a hybrid fashion, but rather natively.

One other main advantage to a native application is the increased functionality available. As the hybrid application utilises multiple PhoneGap API calls, there is somewhat reduced functionality sometimes with the scope of the API calls. In the development of a native application, the full extent of a systems features can be accessed easily.

3.3 Other Cross Platform Frameworks

There are multiple other cross platform frameworks available other than PhoneGap, many of which were examined in the context of this project. Each framework that was examined is detailed below, with reasons as to why it was not further utilised or investigated [28].

3.3.1 Marmalade SDK

Marmalade [29] is a software development kit (SDK) from Idea Works Limited, it contains library files, and the tools required to develop and deploy cross platform. Marmalade is programmed in C++, and contains most of the features PhoneGap does, however there are two main issues to be accounted for. There are problems with the support for running applications in the background, with no clear indication whether they will work or not work. Indeed it is not guaranteed to work but may work, this is a reliability issue that is just not acceptable considering the application that is being developed here.
3.3. Other Cross Platform Frameworks

There second issue is in relation to licensing for the Marmalade SDK, unlike PhoneGap, Marmalade has a proprietary license setup, ensuring that you must pay to use the SDK, this has multiple issues, not least the cost associated with the licensing. There are multiple support levels depending on what license level is opted for, ensuring that good quality support is unavailable in many instances. The types of licenses also restrict the application in various ways, such as revenue limits and platform support limits.

3.3.2 Feed Henry

Feed Henry [30] is yet another mobile application platform, allowing fast development and deployment of applications. Feed Henry is unique in that the system is cloud based, as opposed to other systems where the development is carried out locally on the machine. Feed Henry provides a platform as a service (PaaS), and allows applications to be developed in web languages such as HTML, CSS and JavaScript similar to PhoneGap.

The major issues with Feed Henry are the lack of background support for an application and the lack of support, Feed Henry is however a relative new comer to the market with much scope to expand.

3.3.3 QT Framework

The QT frameworks Code Less, Create More, Deploy Everywhere strategy is very relevant, this is the aim of most cross platform development systems [31]. The QT application was initially developed as a cross platform tool for desktop based machines, with the first release in 1992. QT uses standard C++ for development and provides many of the features required for this application.

The main problem with the QT framework however is the lack of support for mobile applications. The framework is currently being adapted for mobile, and there are currently external ports for Android, Blackberry and iOS. The lack of support at the current time remains an issue with QT being more concentrated on cross platform graphical user interface programs for desktops.

Once again licensing is also a barrier to the effective use of this framework.
3.3.4 RhoMobile

RhoMobile is a open source framework developed by Motorola for cross platform applications [32]. It has a wide range of supported operating systems but suffers a common problem amongst cross platform frameworks, it does not provide support for background running of applications.

Further, there is very limited support or documentation making development challenging at times.

3.3.5 Appcelerator

Titanium Appcelerator is a platform similar to PhoneGap and utilises the same web languages [33]. It was initially introduced in 2008 and has been growing ever since. Unlike PhoneGap, Appcelerator provides its own development environment, based upon the popular Eclipse development environment. Inside this environment is where all the code is written and tested.

The license type is a free license. The major issue in relation to it is the lack of wide support, with support only for iOS and Android, with a beta version of Blackberry recently released.

3.4 Demonstrator Application

To aid in the development of the full client side features, a demonstrator application was created. This demonstrator application was created in a similar fashion to the final application, using the same technologies and design methods. It was created mainly to test the PhoneGap API and the accuracy of the location calls.

3.4.1 The Application

The application was designed to be a simple test of API geo-location. The implementation was a client side only implementation that had no interaction with a server of web resources.

The application was also an early test of on device testing. Geo-location testing in a simulated environment presents particular challenges with regards to simulating the
3.4. Demonstrator Application

exact locations and accuracy. This demonstrator application was an example of both testing fully the geo-location features but also testing the PhoneGap compatibility when loaded to a device and required to run on a device through use of Apple iOS provisioning profiles.

An early version of the application consisted of simple longitude and latitude results; this was however later expanded to include other location attributes to fully test the system, as seen in Figure 3.4b.

3.4.1.1 Functionality

The main aims of the application have been highlighted above already, testing of both the API calls from PhoneGap and on device testing were two of the main aims associated with the application.

Many of the competitors to PhoneGap, listed in section 3.3, were not suitable for this project due to issues surrounding their abilities to run in the background on a device. Running in the background refers to any application that is not the current active window on a mobile system, most mobile operating systems have the ability to run applications in the background with a method of multitasking through using CPU threads.

There are a number of states for applications. An inactive application is one, which has not been launched. An active application refers to one that is being currently used on screen in the foreground, an inactive application refers to an application that is in the foreground but currently not receiving events, applications are generally only in this state briefly during transitions. A suspended application is an application that is in the background but is not currently executing code, an example of this is any application
3.4. Demonstrator Application

that is not shut down correctly but has not been granted the specific permissions of background running required. A background application finally, as above, is one that has the limited ability to execute code while running in the background.

In some instances, background running is restricted, for example in the case of Apple iOS, the background running is restricted to location updates, audio, voice over IP applications and applications that receive regular updates from external accessories. Additionally, the type of background running must be clearly specified in the application for it to operate correctly.

![Application states of operation](image)

The other main aim of the demonstrator was to test the accuracy of the location updates. Being able to receive updates at various times and then plot them on a map to judge accuracy was a major achievement. If a case arises where the location updates were not sufficiently accurate, there is a danger that, in this context, traffic congestion reports could be matched to the wrong locations and be of no use.

Alongside testing the accuracy, the first steps of developing an algorithm for automatic detection were initiated. What can be considered of great importance for an application that will run in a users pocket on the background of their phone is that it has the ability to recognise state changes. For example, if the vehicle was to stop and the
user disembark and begin walking as a pedestrian, it would be wrong to automatically send alerts based on the walking conditions being experienced by the user.

Therefore a large part of this initial testing of the demonstrator application was to analyse the speed of walking of different types of people. The speed of walking would then be taken into account during the design of the algorithm for the main application.

### 3.4.2 Accomplishments

The demonstrator application was in a large way very successful in accomplishing its aims.

It was proven that event updates with regards to the location can be received when the application is running in the background, and that these location updates remain accurate, in the most case to within 10 metres, which is a sufficient level of accuracy to have achieved. The background running proved a major success and one of the key deliverables of this demonstrator application.

This level of accuracy proved that the application would be sufficient to use to capture location and speed updates about the vehicles.

The Time To First Fix (TTFF) can be a major issue in some GPS systems. This TTFF is the time it takes from application start to the application receiving and outputting its first location update. It was found on an iOS PhoneGap application that this time was negligible. The only delay being the application initialising as standard for an application of its type. Once initialisation had completed, the location updates were received almost instantly, showing an extremely small TTFF, which is advantageous also.

An unintended side effect was also shown up in the application, a side effect that had not been covered in any documentation relating to PhoneGap or iOS. This was that location updates are only provided on movement. For example, if the user does not move position, updates to the location will not be provided. If the user moves position enough to register, roughly 5-10 metres, the system will recognise this and update. This unintended consequence is actually an advantage, as it now means that less battery and CPU power will be used if the user is not moving and if the application is still running in the background. Previously it has been envisaged that updates would be received every three seconds.

The walking speed of humans was something that needed to be investigated for the application, as mentioned above to aid in the development of the algorithm. The speed
of walking was found to be generally not greater than 1.2 metres per second and in nearly
every case never larger than 1.5 metres per second. This was tested with various testers
to ensure accuracy and consistency also.

The result of learning the walking speed had an impact on the algorithm to auto-
matically detect congestion which could now discount any movement between 0 and 1.5
metres per second on the account it could be a pedestrian.

Low level of battery and CPU usage is critical to a user continuing to use the appli-
cation. Over 4 hours of running, this demonstrator application did not show to have any
adverse effect on the battery levels beyond standard. The exact battery levels recorded
throughout are shown in section 3.4.3.

3.4.3 Resource Usage

The results of this application were extremely encouraging going forward for the
project. This provided a basis to develop on top of and proved that some of the ma-
jor challenges, which needed to be accomplished, were possible.

Some of the accomplishments are mentioned above. The main results from the appli-
cation are the background running ability, the walking speed and the low level of energy
usage.

Resources usage of the application is very important. When developing in iOS, the
Xcode program provides an instruments tool to log any of the usage by the application.
This can then be compared against a standard control version to evaluate the CPU,
network and battery usage of an application. This application did not utilise the network
at all, and is therefore of no relevance here, but is used in the final application.

The battery usage and CPU are linked closely. As the CPU usage increases, the
battery usage will also increase, therefore to ensure a long battery life it is critical to keep
the CPU low at all times and to not put undue pressure onto it.

Xcode provides the instruments tool which outputs a trace showing the application
usage on a number of these factors, and visually graphs the results.

Below, figure 3.6 shows a trace of the standard usage of a phone when running.
3.4. Demonstrator Application

It can be seen here that there are spikes across all three graphed parameters at certain times. These times correspond to times of activity for the application. There are 4 panes available on this screen, the first is a small pane, across the top displaying a number of flags. The black and white flags signify when an application is brought to the foreground or returned to the background. The blue flags signify when the phone is going into sleep mode or entering active mode. The energy usage is a scale of 0 to 20 corresponding to the level of energy being used. The CPU activity bar is a percentage of the available CPU being utilised, the red section donating the usage by the application in the foreground and the green section donating graphics usage. Finally the display brightness section shows simply whether the display was darkened or bright.

Through this trace it can also be seen how the operating system responds when nothing is running in the background, shutting completely down after a matter of seconds.

When the trace for the demonstrator application, which has background running capability and permission, is viewed the resources can be clearly seen and identified. The trace for the demonstrator application is shown in Figure 3.7, over a 10 minute run time.

It can be clearly seen that there are two main areas of high levels of activity, showing both points where the application was brought to the foreground and sent to the back-
3.4. Demonstrator Application

The bottom indicator did not exist on the previous trace. This is a trace of the GPS activity, simply showing when the GPS is active (red) and when it is inactive (black). This also shows the low Time To First Fix as discussed in section 3.4.2. The display brightness is also as expected with jumps in the areas when the phone was being used and illuminated, which is at the two application use times, the end and once during operation, corresponding to the small jump seen at the two minute mark.

The CPU activity is very interesting for this case. There is the expected spike at the beginning, end and in the middle when the application is being used. This is not unexpected and the total activity during this time is reasonably low, showing good efficiency of the PhoneGap application. In between the spikes however the usage is much higher than previously accounted for. This can be explained however, the blue section of the CPU usage refers to audio processing, as a result of the music player running in parallel in the background. Discounting the blue section there is a small CPU usage maintained throughout, corresponding to the application location updates being registered and the logic being carried out on these.

The most critical aspect of the trace is however the battery usage, or as referred to above, the energy usage. The lower the energy usage the longer the battery will last, as these are linked directly. The energy usage levels can be seen to spike as expected in the areas of application usage, but also small spikes at points in between, these smaller spikes are every time the application registers a location update. Geo Location is known to be quite expensive on the battery of a device, and these small spikes reflect that. The energy usage is low throughout otherwise (generally 1/20).

Averaging across the time period shown and indeed across multiple hours, it can be shown that the demonstrator application on average only uses 1/20 of energy if left running in the background, and extremely low figure that should impact on battery life by less than 5%.

This also proves that non-native applications do not have an adverse negative effect on battery life.
4 Design

In this chapter the system design will be discussed. This system will be segmented into its component parts and presented in detail.

A basic overview of the system is an application that can be downloaded and run on a users smart phone. The supported operating systems are the systems supported by PhoneGap, which is currently iOS, Android, Windows Phone 7, Windows Phone 8, Blackberry, Symbian and Bada [34].

The system is a basic client server model, which is an approach to the networking problems encountered. The client side is the application on a users phone and the server side is a website built upon a MySQL background database. The application processes the speed of movement and decides if an incident alert is to be sent to the server, where it is received by the website and stored in the MySQL database. Incidents can then be viewed by approved users via the website.

![The Client-server model](image)

Figure 4.1: Showing multiple clients interacting with a server.

4.1 Technologies Used

There are a multitude of various technologies used throughout this design to make it successful.
The client side, which is the application on the phone, is developed using the PhoneGap framework as discussed in Chapter 3, this framework allows the developer to develop the application in web languages such as JavaScript, HTML and CSS. Through API calls, PhoneGap then runs the application on the mobile operating system.

An additional framework called JQuery Mobile was also used for the user interface of the application. This is a framework that is based on the popular JQuery framework and deals specifically with user interface elements on mobile devices.

The server side of this project is very important also, as this is how approved organisations and people can log in to view all current incidents and to further report them. The bulk of the server side is programmed using PHP languages, while styling is controlled by HTML and a small section of CSS.

The server side uses the popular Google Maps to be able to view the incidents on a map, the Google Mapping system is currently the most popular free mapping system available and includes many desired features.

4.2 Design Of System

There are many requirements of this design, which are explained below.

The system should be robust, this means that the application and the website should not stop working for any reason. It should have a high fault tolerance. For example, should the user input an invalid password to the website, the system should not crash but instead throw an appropriate error informing the user of what has happened. Likewise if a more serious error occurs the system should inform the user of what has happened. This is very important as without a robust and stable system, users will not want to utilise the system in real world scenarios and will lose trust for it.

The system as a whole should be very easy extensible. This is also one of the reasons for utilising the cross platform development method. Additional features can also be very easily added to make the website extensible with new features and reporting of the incidents. In addition the client side application should be very easy to integrate new features to. This is another powerful feature of PhoneGap; with multiple plug-ins available to make it extra extensible if required.

The scaling of the system beyond testing is of great importance. While during testing there are a very limited number of devices in use and limited traffic to the server, as the
4.2. Design Of System

4.2. Design

As the project grows and more users come on stream, the server side will have to be extremely scalable. If the service gets more requests than it can respond to within a reasonable time frame then something has to be done to correct this. The major issues surrounding scalability often surround around supporting distributed systems, performance and data integrity [35].

The final key design requirement is for the entire system to have a high level of usability associated with it. Usability is one of the most important characteristics for any software, if the software does not have a reasonable simple interface or is not fit for purpose then it is of little value to anyone.

The system design is detailed in the following sections. Here the design is detailed according to its architecture and according to each module.

4.2.1 Client Side

The client side of this system is the actual application that the user can download and run on their device. It is the interface to the entire system that the vast majority of users will have. As such it is very important that it is easy and intuitive to use, and extremely stable. If either of these conditions is not true then there is an extremely good chance that the application will not be utilised. The fewer users that the application has, the less accurate the resulting data that is received becomes. If the data is not accurate organisations will not want to use the system and the whole system will no longer be of any value.

The application is being developed using the earlier discussed PhoneGap platform. The development will be carried out using the standard web languages of HTML, CSS and JavaScript. All of the computation is carried out using JavaScript. PhoneGap provides a lot of support through application programming interface calls to the device.

The application is mainly intended for use by a driver of a moving vehicle. There are a number of implications of this. One of the main implications is safety. It would be extremely unsafe and irresponsible for a driver to have to use the application while moving; this would increase the risk of road traffic collisions but also has legal issues in Ireland. Section 3, Road Traffic Act 2006 [36] outlaws the holding of a mobile phone while driving. Therefore the application will have to be as automated as possible and not require interaction with the driver while the vehicle is in motion.

On start-up of the application it would be ideal to have a screen with information on
4.2. Design Of System

the application, how to use it and how to contact support in case of any trouble occurring where the user does not know how to react or needs support.

To ensure the goal of having a high level of usability and encompassing safety of the user, the user interface has to be kept simple and intuitive. The main idea of the user interface shall be that the user can, at any time, signal a number of incidents that have occurred. These incidents can be signalled with easy to use buttons. There is also an option for the user to select modes of operation. The final feature will be a set of two buttons for the user to interact with, the first button to check the parking spaces available in Dublin City on a real time basis, with the data provided by Dublin City Council. The second button for the user is a feed to the road weather conditions as provided by the National Roads Authority.

The user interface design will be developed with the assistance of a touch optimised web framework by the name of jQuery Mobile [37], which stems from the popular JQuery. JQuery Mobile supports multiple operating systems and is compatible fully with PhoneGap. It provides the ability to design a single graphical interface that has the ability to be utilised across multiple operating systems and screen sizes. The extent of usage of JQuery Mobile in this project is to perform the user interface rendering so as to ensure the user interface maintains consistency throughout multiple devices.

The connection between the application and the server is of great importance. Since the application does all of the data processing and only passes the bare minimum of required information to the server in the form of alerts, it is not necessary to have a constantly open connection. It is however required to have a connection that will work when required. To this extent the connection between the client and the server is via a JavaScript call by the name of XML HTTP. It is an API call used to send HTTP or HTTPS requests to a server and receive a response. If no internet connection is available, an error to that effect will be given to the user, otherwise the information should successfully send.

As has been mentioned throughout this report, the ability for the application to run in the background is a key feature. This feature is intrinsic to the entire design of the client side of the system. To enable background running it is essential that the correct permissions are granted appropriate to the operating systems so that the phone does not enter into a sleep mode once the application is pushed to the background. This background running of the application required a lot of investigation and assurances that location updates could be made from the background.
The design of this application specifies that all processing and computation to detect congestion is carried out on the user phone and simply only traffic incidents are sent to the server. To enable this there has to be rugged algorithms in place so as to detect the traffic correctly and decide when traffic congestion has occurred. This is a difficult task, as different people perceive congestion in various ways. One easy way however to get around this problem of perceiving the congestion is to work with the vehicles speed and the rate of decrease in speed.

There are two algorithms used, one for a user travelling in a car and one for a user travelling in a bus. The reason for the different algorithms is that the bus will stop often and experience jumps and declines in its speed different to a car. The algorithm for a car is to take the previous three speeds, ensure that the speed is actually decreasing, and then ensure that the level the speed is decreasing by is sufficient to safely estimate congestion has occurred in the area. The algorithm will be explained in more detail in the implementation chapter, Chapter 5, together with its code. The algorithm works as it detects the car slowing down and its rate, but does not detect if the car comes to a stop, therefore discounting stops at traffic lights and any other reasons for stopping. The bus algorithm is much the same with varying tolerances.

### 4.2.2 Server Side

The server side of this project has two main sections. The website element catches the incidents from the client and later displays them. The database stores the incidents and handles them accordingly.

#### 4.2.2.1 Website

The website element is an integral part of the project. Without this interface, organisations would be unable to view any occurring incidents.

The website design is such that the site will accept user registrations and contain user accounts, also stored in the MySQL database. Any user who attempts to register will have to confirm their email address by email activation. This is intended for security reasons to ensure a valid user contact. Any user registering will also be required to be approved by an administrator. This is to ensure that only authorised bodies are able to view the information.

The website is required to have a high level of security associated with it. This is
because of the data being stored. The incident data is not sensitive, as it does not correlate back to a specific identifiable user rather just a user number that is changed every 24 hours. However the details of the user accounts, such as username, email and password are sensitive and their security is important. This is also a legal issue under the Data Protection Laws [38, 39].

Security on a web application is a challenge, which is dealt with in greater detail later. It is however of great importance to ensure that the website and corresponding database are developed in a secure manner from the beginning. For example it would not be correct to finish their development and then begin to consider security. The use of user accounts and specific areas available only to users is the simplest method of security that is employed here.

An aim, as with much of the project is to have the website as usable as possible, for similar reasons to the client application, but also to keep support emails required to a minimum.

Figure 4.2: A basic site map for the home page of the website.

The website will provide much of the backbone of this project. The client application will connect to a specific page on the website to input data into the server and all the database interaction will be carried out via the website frontend, with the exception of the administrator who will have root database access. Therefore the reliability of the website is of great importance. The website should be functioning as much as possible, which involves a reliable server infrastructure and good quality testing.

The languages chosen with which to write the website are of obvious importance.
4.2. Design Of System

HTML and CSS are chosen to handle the graphics component of the site and any of the output that can be seen. HTML would however be unable to handle the background processing and database interaction. Therefore the bulk of the website is written in PHP. PHP is a powerful enough language to enable any function that is required in this instance. XML is also required, to display the incidents on a map and also for the custom National Roads Authority weather information that can be accessed via inside the application.

4.2.2.2 Database

The database is the second half of the server implementation. The database is a MySQL database installation. The alternative would be to use an Oracle database. The MySQL database was opted for however due to its usability and documentation.

The database is required to contain two different tables. The first table is a table of incident data, as received from multiple clients. This data must be given a timestamp and an auto generated incident id, which is auto incremented. The database also takes and stores the latitude, longitude, incident type and user id as sent to it.

<table>
<thead>
<tr>
<th>users</th>
<th>incidents</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK</td>
<td>PK</td>
</tr>
<tr>
<td>username</td>
<td>id</td>
</tr>
<tr>
<td>email</td>
<td>incidenttype</td>
</tr>
<tr>
<td>password</td>
<td>latitude</td>
</tr>
<tr>
<td>admin_approval</td>
<td>longitude</td>
</tr>
<tr>
<td>activation</td>
<td>eventtime</td>
</tr>
</tbody>
</table>

Figure 4.3: Database model.

The second table required is the user table, to store the details of the users authorised to use the website. The user table will store the user name and email of the user. It will store an indicator to indicate whether the user has been approved by the administrator and whether they have verified their email address. The password of a user will also be stored in the database, but will be hashed using SHA256. SHA256 provides a 256-bit hash of the password. The result of this is that even if a malicious source was to come across the hash of the password it would be near impossible with current technology to decrypt and work out the password in plaintext within a reasonable timescale.

There are various user accounts for the database also. The root account has full access
and control over the database. There are two other accounts, one of whom has read-only access only and the other has read/write access. The credentials associated with these accounts are at all times hidden from the website user. In addition the access method to the database is always hidden from the website user through the use of various PHP functions.

4.3 Languages And Frameworks

There are multiple languages used throughout this project, all based on web languages. Frameworks used include PhoneGap as discussed and JQuery Mobile.

HTML is used on both the client and server sides. This is utilised as a mark-up language in both incidents to add styling to the intended page. Its use is kept to a minimum and it interacts with the JavaScript and CSS used.

CSS is another styling languages used throughout. The CSS employed is very simple and adds style to text to support the HTML implementation.

SQL is used briefly in the creation and maintenance of the database. It is the standard language for such a task.

XML is yet another mark-up language, this is however used to define a set of rules and the information in the case of plotting the incidents on the map. It is also used to detail the National Roads Authority information and output it for the client application.

JavaScript is the language used for the bulk of the client side application. JavaScript is a scripting language that in this case carries out all the API calls and carries out all of the computation on the client side.

PHP, as mentioned previously is the language of choice for the server side website implemented. This language was chosen because of its powerful nature to complete the task required in this instance.

JQuery Mobile and PhoneGap are two frameworks used. PhoneGap has been dealt with extensively. JQuery Mobile has been dealt with briefly previously. It is a mobile framework for touch optimised devices. In this context it is used to handle the user interface when moving over different operating systems, screen sizes and rotations.
4.4 Technical Approach

Below are three sections detailing the approach taken to three problems.

4.4.1 User Interface

The user interface has been mentioned extensively throughout this report. The need for a good user interface, as mentioned previously also is to ensure that the user will continue to use the system, and that the user interface remains constant throughout the multiple operating systems supported.

There are two elements of user interface in the system, the user interface on the server wide website and the user interface of the application. The main challenge relates to the user interface on the client side.

The user interface must run seamlessly cross platform, on multiple displays sizes and resolutions, and must also be able to handle rotations while the application is running.

For this the aforementioned JQuery Mobile was utilised. This framework was able to provide all the required support and more. It includes many features such as its own graphic design, and custom icons. It is also possible for the framework to work when hosted locally, which is of importance. Additionally, there are no issues with using JQuery Mobile for applications destined for the appropriate store.

The design initially of the application was simple, as shown in the figure 4.4a. While later versions of the design differed, they continued to maintain the basic design and outline. Later versions did however begin to include other features such as additional buttons and options.
4.4. Technical Approach

4.4.2 Background Operation

Another area of great interest is the background operation of the application. This area is also critical to the overall operation of the client side, and has also been discussed in detail earlier.

Since iOS 4, Apple has allowed applications on the platform to multitask, what this means is that applications can run in the background of a phone even after the user switches to another application. They run in the background while preserving as much battery as possible. Most applications are suspended shortly after entering the background. Only designated services can operate indefinitely in the background, these are applications that play audible content while in the background, that use location services, that support Voice over Internet Protocol and applications that receive regular updates.
from external devices. The permissions on other operating systems such as Android and Windows Phone are not as restrictive but still exist.

While location updates in the background have been specified as being allowed, the sending of network traffic in the background had previously been not possible, and is not supported, both in iOS and other operating systems.

This problem was however overcome in this application. By using web languages, the call back function from the allowed location updates also sent network traffic, via a XMLHTTP request. This is that the function called in the event of a success in getting the location is also able to send network updates. This is a very valuable discovery and one which is previously undocumented. It is an unintended but advantageous side effect of using the PhoneGap system, with the resources of web languages. The full code for this implementation can be seen on the attached CD.

4.4.3 Algorithm

The algorithm has thus far been much talked about and is indeed a very important aspect to this project. The main algorithm is a part of the client side application that detects congestion based on the speed of travel of the user.

The algorithm solely uses the speed that the phone is moving at, received through the speed feature of the geo-location API. The algorithm is based on published research that has been previously carried out in this area. This research is all referenced in the Bibliography section [16, 40–51]. Since no project to our knowledge has been carried out with the same scope as this one, i.e. detecting congestion using just speed on one users device, the algorithm uses results from a number of similar works.

The application provides three different user types. There are options for a car user, a bus user and pedestrian user. The application will not monitor when in pedestrian mode. The bus mode is different to the car to take into account the frequent start stop nature of a bus.

For the car user type, the user must be travelling greater than 1.5 metres per seconds, the recorded speed of a brisk walk, as found during testing of the demonstrator application, this ensures the user is still in a moving vehicle and not on foot. The next three IF statements are measuring the rate of descent of the users speed, if it is sufficient and passes each test then an incident alert is sent.
4.4. Technical Approach

Listing 4.1: Car Algorithm

```javascript
speed3 = speed2;
speed2 = speed1;
speed1 = position.coords.speed;
var speedav = ((speed1+speed2+speed3)/3); //Calculates average speed
if (usertype == "car") { //If the usertype is a car
    if (speed1 > 1.5) { //Ensure car is still moving slowly, hasnt stopped at
        lights/parked
            if (speed3 > (speed1+(speed1*0.3))) { //Ensure speed is actually
decreasing, not fluctuating
                if ((speed2*0.65) < (speed1)) { //Ensures level of decent is ok & not
                    emergency stop or very sudden
                        if(speedav <= (1.25*speed3)){ //Check the average speed is lowering
                            fast enough
```

The above shows a code snippet for the car algorithm.

The bus algorithm is much the same but utilises altered figures.

Listing 4.2: Bus Algorithm

```javascript
speed3 = speed2;
speed2 = speed1;
speed1 = position.coords.speed;
if (usertype == "bus") {
    if (speed1 > 1.2 && speed1 < 8) { //If between 5 and 30 km/hr roughly,
        ensures not stopped at stops/lights
            if (speed2 > (speed1*1.25)) { //If decreasing at a good rate
                if((speed2*0.5) < speed1){
```

One of the main undefined areas of this algorithm is cyclists and motorcyclists. It is however true that cyclists and motorcyclists will experience forms of congestion, just not to the same extent as other road users. This is an area for future development work with the application and algorithm.
4.5 Security Considerations

There are many security considerations to be taken into account for this project [52, 53]. Some of the main security considerations involve data protection and data privacy, which is covered legally in Ireland under the Data Protection Act 2003 [38, 39]. This act clearly outlines the responsibilities of a data collector.

Security in any system is very important but that is particularly relevant here. The developer is responsible for ensuring that the data is stored safely and securely, and that attacks on both the data and the system are not possible.

One of the best methods of data security is education, education such as ensuring that a user knows what data they are supplying and how sensitive that data is. In this instance the majority of the data supplied will be supplied automatically by the phone. The only data provided by a user being the username, email and password of somebody attempting to log into the server side website.

There are many varied types of attacks that are possible. Some of these can be classed as malicious offensive attacks while some will be more passive in nature; they all however pose a serious threat to the system as a whole. There is also scope in this instance for innocent attacks to occur whereby a user sends false alerts, sometimes with little to no understanding of the consequences.

4.5.1 Attacks

The OWASP Top 10 is an annual document produced by the Open Web Application Security Project (OWASP) foundation, which represents the top 10 threats for web application security in that particular year. Year on year SQL injection is named in number one position as being of the greatest threat.

A SQL injection attack involves the insertion of a SQL query to a request. A SQL injection exploit can return sensitive user data from the database or modify the database. To perform SQL injection the attacker inserts a different condition in the input in order to effect the execution of predefined SQL commands. For example: 

\[ \text{SELECT * FROM users WHERE username='suser' } \]

could be exploited by setting the suser variable to be 

\[ \text{Suser = anyone OR 'true'=true } \].

Instead of returning a single users password, email or other information as originally intended, all the details in the database would be returned. This attack is of particular
relevance to the servers website implementation and is of little use on the client side.

Luckily knowing of SQL injections existence makes it easy to combat against. By simply escaping any illegal strings and sanitising data any problematic code will not reach the database.

A man in the middle attack is yet another malicious attack, this time targeting the link between the client side and server side. This attack involves looking at network traffic, intercepting it and including malicious code. Malicious code could take any form and potentially break the entire server side of the system. This project is protected against it by connecting using mainly cellular data, which is encrypted by the carrier and not easily accessible. However in a situation whereby a user was on a bus which had a public Wi-Fi network, an attacker could attempt to insert malicious code, however, as with SQL injection, all code is sanitised and cleaned before the database is queried and is therefore safe. Figure 4.5 shows the basic structure of such an attack.

![Figure 4.5: Structure of a man in the middle type attack.](image)

Additionally, the link to add an incident to the server is not publicly available and is hard programmed into the application. This ensures that it should not come under attack by people learning it and flooding the system with new false incidents.

Cross Site Scripting (XSS) is an attack that effects many websites, particularly websites with user inputs such as guest books. It involves being able to plant scripting language code into content that is served on an innocent users page when they open it, either to damage their computer or simply deface the website. This attack is however not relevant to a high extent here with the exception of the contact page, where a mali-
cious user could access this page without being logged in. All data is however escaped of any unwarranted characters before being passed to the mail function for sending to the support email address.

4.5.2 Incorrect Usage

Incorrect usage relates to usage by a user that is not correct, or unintended. This may occur either maliciously or innocently depending on the user. With an application that is publically available to download for free this is a major concern. An example of incorrect usage could be the signalling of multiple events over a short space of time for an incident that has not occurred.

The application provides two main defences against such a problem. User identification numbers are assigned to each application randomly on first install. This user identification number is five digits long and is sent with every alert from the phone. This way if the administrator was to believe a particular user was sending false information, the user could be blocked from adding anything further to the database. Rather simply instead, it would be easily possible to delete any incidents reported from that users user ID.

This system is however full of privacy concerns. Using such a method the site administrator or registered user could possibly track a user over a wide range, and build up a profile of known locations or similar. This is an obvious privacy issue. Therefore the user id is changed every 24 hours by randomly generating a new one, this ensures a reasonable balance between user privacy and security.

As an added counter measure to incorrect usage, there is a feature in place to ensure that no two alerts can be sent within a five-minute period. This should hopefully work to alleviate any problem.

4.5.3 Other

Sessions are used to authenticate users on the website by the user logging in with a username and password. These sessions are set to expire after twenty-four minutes if no activity is detected. This ensures that a user cannot inadvertently leave themselves logged in for a non-authorised user to view data.

A denial of service attack is a real possibility for any web-based application. There
are very limited ways to protect against this for small websites. The possibly best counter measure to this is that the server is hosted on Amazon EC2, a highly scalable host. If a denial of service attack was to be initiated it would be very simple to start more servers in a short time frame to combat this.
5 Implementation

The implementation of any project or system is of huge importance as it is a test of the design work and of the background research carried out.

This project was successfully implemented in its entirety. The client side application was fully implemented in a cross platform manner and achieved all of its aims.

The server side was also fully implemented. It is available at www.trafficapplication.com. The site is hosted on an Amazon EC2 server remotely.

Throughout the implementation and development portion of the project, all of the code was maintained in a Git repository to ensure version control at all times and consistency across multiple development devices.

How the actual design was implemented will be looked at and investigated below, including how the technologies used and how the features were implemented.

5.1 Client Side

The client side application was fully implemented as designed. The application met each of its aims. The application was very successful in running cross platform, in the background and using a minimal amount of energy or network.

The application was, as described later in detail in the testing section fully implemented on device for iOS and for Android. It was further tested in an emulator environment for Windows Phone 7 and for Windows Phone 8. There are minor quirks associated with each implementation that in many cases involves small changes to the code base so as to ensure that the application runs smoothly. These changes are, for the most part very small in nature and the majority related to user interface issues.

The user interface was a huge success also employing the previously discussed Jquery Mobile framework, to ensure that the user interface was truly cross platform and adaptable. The user interface is pictured throughout this report in its many different forms, the forms ranging from a simple portrait view to some more complex scaled tablet devices.

The technology involved in the implementation of the client side was as expected.
5.1. Client Side

HTML, CSS and JavaScript were all utilised heavily.

Some of the main challenges in implementation involved the testing of the application both during and post development. Due to the applications location based nature, it is extremely hard to test it in a static location, as such on many occasions testing had to be delayed for various reasons.

A number of the PhoneGap APIs were used. The main usage involved the location API, and the speed, latitude and longitude attributes for the API. The local storage API was also used as was the notifications API. All of these API calls worked seamlessly when transferred to various devices.

5.1.1 Features

There are many features involved with this application. A brief overview of many of them will now be detailed below, along with appropriate screenshots and code snippets where required.

Any of the features that require user input are handled through using the JavaScript onCall method, which activates when the user interacts with the on screen button.

During start-up when both the application is loading and the PhoneGap framework is loading a basic loading splash screen is displayed. The idea of this is so that the user will be kept informed of what is occurring in relation to the running of the application, should there be any problems.

On starting the application for the first time, there is an information screen displayed. This screen will only show on the first install of the application unless a user presses the information button in the top right hand corner of the screen to call it. The page provides basic information on the application, usage instructions and how to get in contact with support for the application.

There are also a number of checks carried out by the application before it will run fully. There is a check to ensure that the device is running the correct version of the operating system, such that the operating system is tested fully. For example, on iOS, the application will not run in the background below iOS 4, if the user was using a pre iOS 4 system the application would warn the user that it is untested and give the option to exit or continue.

There are two event listeners established also. One is for the event the phone goes offline from an internet connection. If this occurs an error is given with the reason for
The next check to occur is the simple user identification number check. If it is the first install of the application a new number will be generated. If not the old one will be retrieved from memory, and its date checked. If it is older than 24 hours a new user id will be generated accordingly.

Listing 5.1: User ID check and creation

```javascript
function checkuid() { //Checks for a present user id
    uid = window.localStorage.getItem("uid"); //Tries to find if a user id present already
    var t = window.localStorage.getItem("t"); //Gets the time the id was created
    if (uid == null) { //If no id present already
        userIDgeneration(); //Creates new one
        return;
    }
    else {
        var p = new Date(); //Gets the time & date
        var k = new Date(t); //Converts t to the time and date format
        if (k < p) { //Gets the date and time, if the user id is older than 24 hours it is expired and new one made
            window.localStorage.removeItem("uid"); //Remove the stored uid value
            userIDgeneration(); //Function for creating new id
            return;
        }
    }
}
```

Figure 5.1: The Splash Screen on Loading  Figure 5.2: Start Up Information Screen

the error. Likewise an event listener listens for the return of an internet connection.
5.1. Client Side

```javascript
else { // Otherwise the id is equal to what is stored
    uid = window.localStorage.getItem("uid");
    return;
}
}

function userIDgeneration() { // Generates new user id
    var oneday = new Date();
    oneday.setHours(oneday.getHours() + 24); // The expiry time of the id, 
    the current time plus 24 hours
    uid = Math.floor(Math.random() * 1111111); // Random number
    window.localStorage.setItem("uid", uid); // Store random number
    window.localStorage.setItem("t", oneday); // Store expiry time
    return;
}
```

The automatic alerts are then enabled, and active unless the user presses the Stop 
Watch button so as to stop watching the location and disable automatic alerts.

Manual alerts can also be signalled at any time. These alerts are to alert to a traffic 
accident, a traffic light fault, and traffic congestion if the user believes it may not have 
been automatically picked up by the system.

![Figure 5.3: Notification Sent Confirmation](image)

The important modes of operation are enabled on start-up by selecting the car user 
type by default. A user is free to change this without problem. For the first time of
pressing pedestrian mode the system will inform the user that no monitoring takes place in pedestrian mode.

There is a block on alerts sent from the phone to ensure that alerts are not sent more frequently than five minutes apart. If a user attempts to send an alert within five minutes of sending a previous alert they are given an error to notify them of the problem. This relates to a security issue of incorrect usage, as discussed in section 4.5.2.

![Figure 5.4: Timeout Notification](image)

Should the user be unaware of the alerts that the phone is sending and wants to check them; there is a facility to do so. In the top left corner of the main screen there is a button with the ability to check the last 5 alerts sent from the phone. These alerts are always ranked in order with the most recent sent alert at the top.

![Figure 5.5: Last 5 Alerts Pages](image)
The final two of the main features of the client side application are both features of particular use to the user. The first is a button to connect to a live feed of parking in Dublin, as provided by Dublin City Council. Within this page there is also a link for users to link into seeing the traffic cameras for the Dublin area.

![Dublin Parking Information](image)

**Figure 5.6: Dublin Parking Information**

The other button provides a link to a page that is generated on the server. The page that is generated on the server serves a table form of the National Roads Authority road weather live XML feed. In addition, should the temperature on the roads drop below -3 Celsius when the user is using the application, an alert will be displayed, provided an alert has not been issued in the previous six hours.
5.2 Server Side

Listed above are the main features of the application, which affect the user and the general operation. Should the code for any of these be required to investigate the implementation, it is included on CD with this project.

The HTML and CSS code remains relatively simple throughout, as a result of the JQuery Mobile implementation, using defined buttons, themes and icons from the JQuery Mobile library. The HTML code provides a simple route of access to the background JavaScript code.

5.2 Server Side

The server side to this project is a two-part implementation. As discussed earlier in the design section, there is both a website where the user interaction takes place and a background MySQL database which records incidents and user accounts.

The server side is hosted on Amazon EC2 service, meaning it is hosted in the cloud, with a near 100% availability rate, and huge range for easy scaling of the system. With the Amazon Web Service, a user is able to boot a virtual machine almost instantly, and likewise shut down virtual machines to give increased control over site load management and security.

Many of the challenges of the server side development originated from the poor error reporting of PHP, for example if a mistake is made of a semi colon being left out, the web page will not serve up anything and the only way to troubleshoot is to begin to crawl through the web pages entire code.
5.2.1 Website

The website by the most part is written in PHP, as discussed in the design section. The website is highly functional and secure also. The graphical section of the website is handled from two files, one file to cover the top half of the HTML output and a second file to cover the bottom half of the HTML output. These are added at the beginning and end of every PHP file. This ensures that the PHP files remain de-cluttered but also that the graphical aspect of the site is easily altered from within two files rather than working across multiples.

The server code is very segregated. There are twenty-nine various files involved in all aspects of the website. There are files that are directly accessible, and there are many files that are helper files which the user does not have direct access to or knowledge about.

As has been mentioned throughout the code, an aim of the project is to ensure that the site is secure. The implementation was developed at all times with this aim in mind. Following the implementation a static analysis code check was carried out on the website using a tool called PHP Code Sniffer, this tool showed zero issues with the code.

The process throughout development involved the slow addition of features to the site. The site began as a simple site with the ability to add incidents to the database and to view the incidents that were in the database in a table format. Following this more features were added continuously and built around and on top of the existing features.

When a user browses to the website, if they manage to land on a page that is undefined, a custom 404 error page (Figure 5.9) will load and show as opposed to the standard browser 404 error. This error page will then redirect to the home screen, where the user is required to log in.

![Image](Image)

Figure 5.8: Home Screen of Website.

On the home page some basic information is displayed. If the user has a username and
password, the user can log in. If not there is the ability to either change the password by resetting it or create a new user account. If the user creates a new user account they are redirected. The username, email and password is required, the password must be between 6 and 16 characters long, and entered twice to verify. Following this the user will receive an email with a link to an activation page so as to activate their email address. Likewise the administrator will receive a link to activate the user as a genuine approved user. Not until both these tasks are completed will the user be able to login.

Had the user forgotten the password and chosen that option. They would enter their username to reset the password; a random word can be generated and combined with a random number from 1 to 100. The new password will then be emailed to the users verified email address.

If however the user had of known their password, they could have simply logged in.
5.2. Server Side

Once they log in once, a session is created using the PHP session value; they are then logged into the whole site. They will be logged out when they chose to or once they have 24 minutes of inactivity.

Once logged in, the user has options to change the password, view a table of incidents or view a map of incidents. When viewing either the table of incidents or the map of incidents any incidents older than three hours will be deleted from the database as they can be assumed to be no longer relevant. If the incident is still on going but has been deleted it is likely it will be re highlighted as an incident soon or has already by a nearby user.

For a user to contact the site, there is a contact page. To use this page the user does not have to be signed in, therefore any site visitor is free to use this function to send an email to the site.

![Figure 5.12: Incidents Shown in table form.](image)

The page that the application connects to in order to insert incidents to the database is also based on the website. This page simply takes the parameters as GET parameters from the URL called for by the application. It then inserts the parameters into the database as long as there are no clashes or errors.

Google Maps provide much of the implementation required in the mapping function of the project. The map is based on Google Maps and a Google Maps implementation but is customised heavily to cater for the project. The map page can only be viewed by a logged in and approved user. The incidents are taken from the database and populated into a XML file. From this XML file the incidents can then be correctly read into and
displayed on the mapping system. To display the map the XML file is loaded, and the markers on the map mapped to the latitude and longitude, the corresponding text for the marker is rendered including the appropriate icon, which has been defined depending on the incident type. Shown below in figure 5.13 is a screenshot of the map showing a number of current active incidents. Similar to the table, incidents will get deleted when they are three hours old.

![Figure 5.13: Incidents Shown in map form.](image)

Above is a small breakdown of some of the main functions and implementations of the server side website. For a full overview, please feel free to visit the site and register. Additionally the full source code is provided on the appendix CD.

### 5.2.2 Database

The function of the database is to store all of the incident information and the information relating to the user accounts of the website. The database has the ability to be queried in order to carry out these tasks.

A MySQL 5.5.29 database was used for this system. The database has two tables. The storage engine used is the default storage engine, InnoDB, its supports are more than
sufficient for this projects requirements. The SQL code for the database implementation is show below.

Listing 5.2: Database Code

```
CREATE TABLE IF NOT EXISTS 'incidents' (
    'id' int(11) unsigned NOT NULL AUTO_INCREMENT,
    'incidenttype' varchar(30) NOT NULL,
    'latitude' varchar(45) NOT NULL,
    'longitude' varchar(45) NOT NULL,
    'eventtime' timestamp NULL DEFAULT CURRENT_TIMESTAMP,
    'uid' varchar(10) DEFAULT NULL,
    PRIMARY KEY ('id'),
    KEY 'time' ('eventtime')
) )

CREATE TABLE IF NOT EXISTS 'user' (
    'username' varchar(16) NOT NULL,
    'password' char(64) NOT NULL,
    'email' varchar(100) NOT NULL,
    'admin_approval' varchar(40) NOT NULL DEFAULT '1',
    'activation' varchar(40) NOT NULL DEFAULT '1',
    PRIMARY KEY ('username')
) )
```

The database is administered using the phpMyAdmin tool, a free and open source tool written in PHP to handle the MySQL administration. Through the phpMyAdmin portal the database user can create, modify, delete and alter tables as they please, according to the set of permissions surrounding that particular user.

As in all sections previous, security is of importance in this instance once again. This is where any sensitive data is stored, in particular a users password, which is hashed using Sha256. An example user is shown beneath showing the hash.

![Figure 5.14: User details stored in database.](image)
5.3 Implementation Issues

This hash is reasonably secure in that it is relatively useless to any potential hacker with current technology and processes.

5.3 Implementation Issues

There were a number of issues related to the implementation of the project. Throughout the process there were multiple unexpected issues that occurred. Luckily many of these were easily solved with a work around proving possible. In other cases, planned features had to be cancelled.

One planned feature that was cancelled was the planned introduction of a motion detector via the accelerometer. This was planned to automatically detect moments of heavy breaking. However there is no API access to the accelerometer when the application does not run in the foreground, therefore this plan was cancelled as it would have been completely under-utilised and ineffective.

5.3.1 Permissions Issue

A specific area of issues that arose multiple times over the course of the project was the area of permissions. The application required multiple permissions to run successfully, both PhoneGap permissions and operating system specific permissions.

All external web addresses are required to be specified on a whitelist, for security reasons within PhoneGap. This was a large issue on numerous occasions and explained many of the errors received at times. For example, the application could not connect to the incident entry page, which required a lot of troubleshooting, as this was the first time the particular error had occurred.

Background running of the application has been talked about a lot in this report and was a pivotal part of the project. There were issues relating to the background operation of geo location on specific operating systems, which has been dealt with elsewhere in this report. This issue remained both a serious issue and concern throughout the project but has been successfully resolved with background operation a success.
5.3. Implementation Issues

5.3.2 Technological Issues

The web languages used for the implementation of this project proved extremely difficult to work with at times. There is an extremely poor error reporting system with each of them, leaving out a simple semi colon can break the entire page. This proved extremely testing at times to locate and fix errors when they occurred.

The accuracy of the location services on a particular device proved problematic at times during testing. If the location was not accurate enough problems have been caused at times with the algorithm for automatic detection, as the speed results received were inaccurate. Luckily, this is a rare occurrence.

The implementation of Jquery Mobile that is utilised has issues associated with it and using custom icons, such as those used for the manual incident buttons. The issue related to the custom images not being shown correctly when the Jquery code was hosted on device rather than a link to a central server. Having examined the CSS aspect to the code it was discovered that the issue surrounded the specifying of the dimensions. This was rectified and a report of an issue sent to the Jquery Mobile developers.
6 Evaluation

Testing is essential for any project of this size before it is released for general use. While testing is carried out regularly throughout on any features that are added and at regular intervals, this does not replace a full testing process upon completion.

Aside from the regular testing carried out, the bulk of testing was carried out in a full test period once the project was completed. The aim of this test period was to not only test for bugs but to also fully test the applications algorithm, operation and background running.

The length of the testing period is not a figure that can easily be defined, however full testing was carried out during a 4-week period, which also included bug fixes and the addition of one extra feature (The last alerts feature), which was deemed necessary during the period. The testing period involved both server side and client side testing.

The client side testing involved the demonstrator application, as detailed in chapter 4, testing of the application on the emulator and on device testing across multiple devices.

The server side testing included testing of the website to ensure functionality was correct, a major part of this testing ensuring that the security protocols were correct, and that users did not have access to areas they should not have. This included testing against problems such as SQL injection, as discussed earlier.

The database was equally tested; to ensure that its limits were as required, and that the user account details were securely stored and maintained. The database was also tested to ensure that saving of incidents could occur correctly from various operational states.

6.1 Testing Method

The bulk of application development was carried out on the iOS platform using Xcode, this was chosen as it is a simple process to test location features using the iOS simulation tool, which can simulate numerous predefined location updates, which have been pre loaded such as a freeway drive, a cycle, a run and a walk. No other software development kit or emulator provides such a service. This made it possible to test regularly throughout
6.1. Testing Method

that the connection was still active between the client and the server, and that alerts and incidents were being correctly sent and received at all times.

Testing during development was constant on the emulator and would occur on addition of each new feature to ensure there was no adverse effect on other features or the operation of the whole application. This testing was an integral part of the development process.

The testing was carried out in a number of phases. The first phase involved statically testing the client side application and the server side implementation and noting any faults or failures with the design. This was carried out on device in the case of the client side and any bugs addressed as they were discovered.

The testing for both the client side and the server side is detailed below, with a comprehensive look at how each was tested.

6.1.1 Client Side

For the client side implementation, a number of different devices were tested to ensure the cross platform aspect of this project was accomplished.

As is mentioned above, testing was performed in a number of different phases. The initial phase involved static testing and was carried out on and iOS version 6 based iPhone. This testing was to ensure there were no bugs that had not been identified during testing on the emulator. Any bugs found during this testing phase were fixed before moving on.

The next phase took place once again on the iOS device; this involved checking to ensure that the background running of the application was operating successfully. To do this the algorithm for traffic congestion was removed completely from the application so that an incident alert would be sent every time the location call back function was called. This allowed the application to be put into the background and to be sure that alerts were still being sent from the background. The reason for the removing of the algorithm was to ensure the correct operation of the background running including sending alerts, as according to iOS documentation, alerts should not have been able to be sent. This was however proved to be wrong for a PhoneGap implementation by the way in which the application sent constant alerts to the server, even when running in the background over long periods of time.

The final phase of testing was the testing of the algorithm to detect congestion automatically. This period of testing was two fold. The algorithm for testing on a bus is slightly different to the algorithm for testing on a car, due to the bus being required to
frequently stop at pick up points.

To test the bus algorithm, it was a simple task of being a passenger on a bus while observing the actual congestion points, in the opinion of the tester. This information was then checked against the generated incidents. The problems with this method are the relative different perceptions of congestion between people. For an example, a bus driver would perceive differently to a user who is un-familiar with the bus route. Therefore it was essential to try to remain impartial during this process. Following the first day of testing it was necessary to alter the algorithm slightly to improve accuracy, the test was then repeated.

In a similar fashion the car algorithm was tested. The car algorithm testing was more efficient and substantial because the tester has the ability to mimic congestion by the manner of driving, as opposed to the bus testing and having to wait for true congestion. This mimicking of congestion was only possible given the knowledge of the underlying algorithm. This technique was only used in early testing to refine the algorithm, following on from early testing the same method as the bus was used, by driving around during rush hour noting congestion locations.

The testing that took place on iOS was by far the most comprehensive, due to all the early testing for bugs being carried out on device on this platform. Additionally since most of the development was carried out in Xcode tailored for iOS no abnormal quirks were noticed. The testing also showed that for the platform, the application was operational in both landscape and portrait modes.

One final piece of testing was carried out while on the iOS device, this involved battery and network usage testing using the instruments tool to log the data, the results of which are described below in section 6.3.

The testing on iOS showed to be a great success with no outstanding bugs known on this platform.

The application was also successfully tested on the Android Emulator throughout the project, at times after a number of features had been implemented for example. This also shows the way in which the application can be easily ported between devices, by simply copying and pasting the main WWW folder.

The Android device testing was carried out on two devices. The first device, a Samsung GT-S5570 was running Android 2.1. The application ran successfully on this device. It was also tested on a Google Nexus 7 tablet running Android 4.0, again there was no issue running on this platform, the display also scaled successfully without bugs to the
6.1. Testing Method  

6. Evaluation

(a) Application Screenshot - Portrait iOS

(b) Applications Screenshot - Landscape iOS

Figure 6.1: Screenshots of Application on iOS

The Android device was also tested independently of the iOS testing, so as to ensure that the background running was still operating correctly and did not require modifications. The testing showed no quirks related to the Android platform once the value for the operating system check was changed to an Android relevant value, chosen as version 2.

The fact that the application was successfully run on more than one Android device, and more than one style, with the Nexus 7 being a tablet device shows the successful cross platform nature of the application once again.

(a) Application Screenshot - Portrait Android

(b) Applications Screenshot - Landscape Android

Figure 6.2: Screenshots of Application on Android

The application has clearly demonstrated that it is cross platform across multiple devices. One further test was carried out. The application was also testing on the Windows Phone 7 and Windows Phone 8 simulators; it was however not tested on a
6.1. Testing Method

Testing on the simulator does not provide full on device testing, but it is rare that errors occur on device testing that have not shown up in testing in the simulated environment. Testing on the simulator proved successful with minor alterations to remove the clause in relation to checking the version number of the operating system removed. With this taken away the application worked perfectly and as required.

Once again this clearly demonstrates the successful cross platform nature of the application, and the success of PhoneGap and JQuery Mobile to make this possible.

![Application Screenshot - Windows Phone](a) Application Screenshot - Windows Phone 7

![Applications Screenshot - Windows Phone 7 View 2](b) Applications Screenshot - Windows Phone 7 View 2

Figure 6.3: Screenshots of Application on Windows Phone 7

6.1.2 Server Side

The server consists of two main parts. The website is where the user interaction takes place. The database is where all website users and incidents are stored.

The testing of the website was also multi faceted. The majority of the testing was carried out when the website was hosted locally, before being uploaded to the web hosted version, as bugs were easier to fix. The website is mainly written in PHP, therefore the bulk of the testing was also carried out on the bare PHP, before any graphics were added to it.

The website performs a number of functions, each of which had to be tested. There were multiple forms of testing. The initial testing, as described above involved testing for small bugs, which mainly related to broken links and items not displaying correctly. All of these were fixed before the graphical side of the website was added. Additionally,
6.1. Testing Method

After the HTML graphics were added many of these had to be rechecked.

When the site was uploaded onto the server, the email system for user registrations and password resets could be properly tested as this had not been looked at previously. This is a very important aspect as it verifies a users email address and also provides a method to reset passwords.

Access control is a major aspect to the website. An administrator-approved user has access to much more features than one who is not. Therefore it was important to test this aspect. To first off ensure that pages were inaccessible for people not correctly logged in, and to ensure that pages were inaccessible to users not yet approved. Access control is of obvious importance as without it the data is free to be viewed by anybody, and the need for user accounts evaporates.

The final, and potentially most important testing was security testing. The security considerations for the website have been detailed in section 4.5, however the testing of these was vital. It is very important that the database is secure at all times, with no access to user details or the hashed passwords, likewise incidents should only be viewable by authorised people.

This security testing also involved penetration testing the website, using techniques such as sniffing for open ports, SQL injection and scripting attacks to attempt to undermine the database and user accounts. Due to the way in which sessions are managed there are no known problems in relation to cross-site forgery requests (CSFR). During testing, The OWASP Top 10 2013 [54] list of threats was used as a basis. The security testing was one of the most important aspects of the testing.

To complement this manual testing. A static analysis was carried out on the code base using PHP Code Sniffer, a static analysis is a method of debugging by examining code without running anything. PHP Code Sniffer is a script that aims to detect violations of defined coding standard, and aims to report on any security issues encountered that may not be picked up in testing. The static code analysis showed no issues with the code base provided. This does not of course prove that the website is secure, but does ensure there are no major issues with the code that have not yet come to light.

As has been mentioned in the previous section, database testing was also carried out. This testing focused more on ensuring that users had the correct access level for their requirement and that there were no entry or modification problems. The testing for this was mainly done on a manual basis with SQL commands designed to test the database, such as searching for incidents out of range and attempting to change a users password.
or details without permission. The database testing showed no obvious or known issues and has been successfully operating with the website.

### 6.2 Testing Issues

Multiple minor issues appeared throughout the testing of the application and the website. These were fixed on a rolling basis as they were discovered. At the time of publication of this report, there are no known bugs throughout the code base, on either client side or server side. The majority of the bugs found during testing related to loops and alerts that had been left in since development time but were easily removed.

There is an issue in relation to running a Windows Phone Simulator on a Apple MAC operated system within a virtual machine. This error is caused across all virtual machines and as such the Windows Phone 7 will not run. The Windows Phone 7 Emulator is itself a virtual machine, and very few virtual machines will host another virtual infrastructure.

![Figure 6.4: The Error Shown on VMware for Windows Phone 7.](image)
To be able to run the emulator for the Windows Phone, any simulation and testing must be carried out on a native Windows machine. As the above image shows, an emulator will start but will be unable to process the required graphics and simply display a white screen due to graphics errors. This is a known issue for the Windows Phone Software Development Kit. This problem also clearly highlights one of the disadvantages of PhoneGap, requiring a large range of integrated development environments to operate correctly.

It is worth noting that VMWare5, a popular virtual machine brand has recently began to support running Hyper-V guest virtual machines with the Windows Phone 8 emulator, allowing Windows Phone 8 development, at the time of publication however Windows Phone 7 Emulator will not operate in a virtual machine [55].

The bulk of the development was carried out in Xcode and in Eclipse, switching between the two for testing purposes. Eclipse provided better error reporting and monitoring than Eclipse does, However Xcode provides support for sending automatic location updates to the simulator, with pre installed routes such as a city walk, city run, and a freeway drive programmed in the San Francisco area which is a great aid when testing a geo location based application.
6.3 Testing Results

The results of the testing process were very encouraging. As has been mentioned above, there are currently no outstanding known bugs on any platform or installation of the device. This is extremely good news and relates back to the aim of having a stable release of the project, to ensure usability by first time users.

The project is also very successful in a cross platform manner, as also outlined above.

The server side website and database is as required, with a mapping interface to view incidents, and a secure and reliable platform for users to interact with.

The results showed that, during the testing period, a large amount of the automatically reported incidents were truly traffic congestion having occurred. This was both an encouraging and discouraging result. Throughout testing it is seen that roughly 25% of the results are in some shape false positives, meaning that the application signalled for congestion correctly 75% of the time. This is a good result, in that 75% of the time, the congestion was recorded correctly and acted upon. The 25% figure is however too high to have trust in the majority of incidents received to the server. This is where the future work of feedback from a wider base of users would be necessary to further refine the algorithm.

The final aim of the project and result was in relation to being non intrusive and utilising minimal resources. As with the demonstrator application, detailed in chapter 4, the energy, CPU and network usage of the application was monitored while in standard operation. This trace of the data is shown below.
There are a number of interesting points to be noted in relation to these graphs. The red section of the sleep/wake simply shows that at all times the phone was powered on. The display brightness section displays when the brightness was high or low, which corresponds to when the application was being utilised in the foreground and for when it was in the background with the screen locked.

The CPU activity can be explained as green being the graphics activity, red being the foreground application activity and grey being the overall activity. This demonstrates that there is a low level of activity when the application is running in the background and a significantly higher level when the phone is in the foreground and being utilised, and could be expected. It can be measured that there is roughly a 10% increase in CPU activity on interaction with the application, when the application is already opened in the foreground with the screen bright.

The energy usage of the application is a very important area, a user will not use the application if it is using a disproportionate amount of energy on the phone. It can be seen that the energy usage is high as expected when the application is being used in the foreground, however, when the application is being used in the background it is also seen that the usage is extremely low in general, with one spike, corresponding to the time in which an automatic alert was sent. The instruments provide a 0-20 scale of energy usage, it is seen that the energy usage for the application not sending notifications in the background is 1/20, while the application running in the foreground has an energy usage of 15/20, which is an expected benchmarking result.

Therefore it is estimated that if the application was left running in the background of
6.3. Testing Results

a user’s phone, the application would consume roughly 6% - 7% battery life in both the background monitoring and sending of incidents.

The network activity of the application is also shown in this instance connected over the cellular network rather than a Wifi network. This is also a very important parameter to be considered. The sending of a notification is shown to use in total roughly 500 bytes outward, and receives roughly 70 bytes from the server. Finding the road information requires sending of 850 bytes, and receiving of 14,000 bytes. Finally the parking information requires 2500 bytes to be sent and 2500 to be received. The difference between the roads information output and the parking output is due to the different hosting sites and corresponding variance in links. These network figures are also very low and therefore use minimum data for of the phone, which is extremely good news for any user.

These results are extremely encouraging and adequately confirm the aims of the project were successful.
7 Conclusions

There are many conclusions to be drawn from the success of this project.

It is shown that cross platform applications are very possible and an area of huge excitement and development in the near future. The cross platform applications do exhibit troubles however, many which were found during the course of this project. It is anticipated that many of these will be solved in the near future with increasing levels of support for PhoneGap and increasing range of API calls.

The expanding range of smartphone operating systems is increasingly making cross platform application development more important, with this trend looking set to continue as more operating systems attempt to enter the market, and smart phones become increasingly popular.

A major requirement for this application to be successful was having the application running in the background being non intrusive to the user by not disrupting the phones other activities, and by using minimal resources such as battery and internet data. It was also found possible to send the alerts in the phone background, something that is not possible on a native iOS application, due to permissions as discussed earlier. This was also a major success and further helped the application to be as non intrusive as was possible.

As discussed in chapter 2, many of the applications and websites currently available for real time traffic information use a wide range of crowd sourced data to estimate the traffic conditions in an area, an area which is pre programmed and thus not easily scalable, this application has proven that it is in fact possible to detect congestion conditions based on speed alone and on the users device alone, rather than having to compile the location and speed details from multiple devices.

Overall this application completed what it set out to, the client side application has been successfully backed up by the server side implementation, is currently fully active and currently working correctly.
7.1 Future Work

There is a lot of future work that can potentially be done in this area to build on the current application.

As with any application, further development can be expected based on user feedback. The feedback provided by members of the public would be useful for improving the design of the application and the features that the application has. The user feedback will also help to fix any possible remaining bugs that may have not been noticed during the testing period.

The algorithm for automatic detection has operated correctly but has many flaws, as discussed in the previous chapter on the results. It is envisaged that this algorithm could be constantly adapted based on the feedback from users and from the organisations online. For example it may be that the threshold for reporting a congestion incident is too low and incidents are occurring too easily, likewise the opposite could be true. With this valuable feedback the algorithm can be adapted to attempt to lower the number of false positive results that are occurring and have been occurring during the testing process.

The application is extremely easily scalable and theoretically will work in any city or in any area. It is however designed with a view to working in the Dublin area initially, based on the client side view with options for Irish traffic conditions and Dublin parking spaces, to move the application to other cities, it is extremely simple to change these buttons to city appropriate content. Likewise the map on the server side automatically views Dublin, something equally easily changed.

One final area of growth for an application that runs in a manner such as this is the possible integration into the background of another application. The potential ability to make this application an easily transferrable module has huge potential. As an example a radio station could have it in the background of their application which would provide an easier way to distribute the application and to ensure that more quality data is received back. Likewise the organisers of a major event where traffic disruption is anticipated could implement this traffic management application as a small module into their event application that could provide tailored traffic control details for their event. Traffic control is a major part of any event management plan [56] and an application such as this could help to control the traffic and give an overview to event organisers of traffic conditions surrounding the venue.
7.1. Future Work

New features can be continuously added also to make the application more attractive to potential users. These features can be tailored to the user and the area, with potential to get more interaction between the application and the user to ensure that the application is regularly started and used.

Before being fully distributed to the public via application stores, it would also be important to minimise the codebase. Currently the code is well commented and formatted, with the application being 7.22Mb in size. By deleting the comments and minimising the free white space, the download size will be decreased and the efficiency of the code will also increase slightly.

The server side is also an area that could feature some work in the future. Aside from any graphical design work to improve the visual impact of the website, work could include the addition of various user types, the addition of new features which could be potentially used to graph previous incidents or provide historical reports on incidents and associated data such as peak days, times and the weather conditions they occur within. Data generated such as this could prove extremely useful to the organisations in advising drivers of when to travel or about the conditions to expect during their travels.

Over time, features allowing the viewing of incidents from a certain geographical area or from a certain specified users could be made possible. Likewise the blocking of a particular users updates unsuitable to a particular organisations need could occur. This would be separate than an administrator blocking a certain users updates completely in a case that they are believed to be fraudulent.

Gerry Cahill of Dublin City FM has been contacted with information in relation to the completed version of the application. Currently, feedback is being awaited on any possible implementation in conjunction with Dublin City FM and Dublin City Council. The completed project information has also been sent to AA RoadWatch for comment but as of publication no response has been received.

The database could have extra features added also, such as a feature to ensure the saving of any incident ever dealt with. This is a simple addition that could add the incident to a history table as well as the current table. It can be anticipated that this table would not be deleted after the three-hour window similar to how the current incidents are deleted. This history table would provide the basis for many of the reports as described above.

The future work on this project is an ever-expanding area, and will be expected to change with varying requirements, user feedback and work done by others in this area.
Bibliography


Electronic Resources

The CD accompanying this Final Year Project Report includes all relevant project code, the report in PDF format and a PDF file explaining the contents of the CD.

Appendices

A Application Information Page

The following is the text of the information page available to the user.
File name is Info.html.
User Guide:

This application runs automatically in the background of your phone. Please ensure you have enabled location services permission, as without this the application cannot run correctly.

On start-up the app begins to watch for signs of traffic congestion. You can stop this process at any time by clicking the Stop Watch button. It can be restarted with the Start Watch button.

In addition a number of events can be flagged. For example, the user can report that heavy traffic is being experienced through the Congestion button, that a traffic accident has occurred in the area through the Accident button and that a traffic light fault has occurred through the Light Fault Button.

For commuters, there are quick reference buttons which link to both the national road conditions on major routes, and the parking function showing the available parking spaces in major Dublin car parks.

There are three modes of operation. The car, bus and pedestrian modes are all different, with the application not monitoring while in pedestrian mode as this would be unrealistic.
To contact, simply email info@trafficapplication.com or press the button at the bottom of the screen.

This application will not work without Internet connection; it will resume automatically on reconnection to the internet if the connection is lost.

These instructions will not show again unless prompted to using the button in the top right corner of the application.

By using this application the user further agrees for their location updates to be transmitted over the internet to a central server for viewing by administrators or selected parties. These locations updates will remain anonymous at all times with user identification numbers being changed every 24 hours to ensure privacy. The application uses a minimal amount of battery and a minimum amount of bandwidth during transmission.

For a full list of terms and conditions or further information simply email using the below button.
B Change Log

The following is the changes sent to version control over the course of the project for the client side. Only major changes were sent, comments can at times include multiple changes also.

29: Minor UI fixes.
28: Minor fixes.
27: Deleted unused files.
26: Updated Comments.
25: Changed Algorithms
24: Fixed last5 alerts bug.
23: Comments added.
22: Last 5 alerts function.
21: Added NRA functions.
20: Removed Accelerometer refs and changed error alerts.
19: ALL UI & bugs.
18: User ID implementation
17: Incident timeblock
16: UI fully completed.
15: Added Icons & SplashScreen
14: Update .gitignore.
13: Added Dublin Parking
12: Added Information Page
11: Background working activated.
10: Gitignore update.
9: Minor changes.
Appendix B.

B. Change Log

8: Revert "Gitignore update".
7: Gitignore update.
6: Minor Changes.
5: Basic Location Monitoring.
4: Network Access
3: Basic Setup.
2: PhoneGap Setup.
1: Initial commit.