Developing an iPhone application to assist persons of reduced mobility around Aberystwyth University sites and buildings

Final Report for CS39440 Major Project

Author: James Bowcott (jab41@aber.ac.uk)
Supervisor: Myra Wilson (mxw@aber.ac.uk)

7th May 2015

Version 1.0 (Final)

This report is submitted as partial fulfilment of a BSc degree in Computer Science with Integrated Industrial Year (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature …… (James Bowcott)

Date ……

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature …… (James Bowcott)

Date ……

Ethics Form Application Number
The Ethics Form Application Number for this project is: 1025.

Student Number
110102160

Abstract

The aim of this project is to create an iPhone application that can assist people of reduced mobility in navigating around Aberystwyth University's campuses and discovering services and facilities available to them.
An initiative called AccessAber already exists to help make the University's buildings and services more accessible via a web app which can be used on any modern mobile device browser.
The purpose of developing a native iOS application is to see what benefits can be gained through a native application which can't be done through a web app. This report details the research and evaluation into the requirements of the target audience for the app, the technologies which are available on the native iPhone platform and what conclusions were reached which defined the final features of the application. The design and implementation of the app are detailed, including what datasets are used, the major data structures within the app and some of the more interesting implementation details, like algorithms.

Contents

1. Background, Analysis & Process

1.1. Background

Aberystwyth University have been looking into ways to improve the accessibility of it's campuses in Aberystwyth (at Penglais and Llanbadarn Fawr), particularly to people of reduced mobility: people who may rely on wheelchairs, crutches or who otherwise have limited ability to walk. The geography of the Penglais campus in particular presents challenges to such people. The steep terrain that the campus is built on means that there are a large about of steep inclines across the site which most people have to negotiate with several times a day to get to and from key buildings. Furthermore, many of the major routes have steps, which obviously cannot be used by wheelchair users and would be of great inconvenience to people using walking aids or who have difficulty climbing stairs.

There are services available to people of reduced mobility to help travel through buildings and sites, such as lifts (traditional and wheelchair lifts adjacent stairs) and automatic doors, but often these services are not easily discoverable to visitors or new students. Furthermore the Penglais campus, being built on a steep hill over gradually over 50 years, has a complicated layout of buildings, roads, paths and stairs, which many people find daunting and confusing for anyone unfamiliar with the site, regardless of whether they have are of reduced mobility or not.

The current services and materials available to people who need help navigating around the campuses are very limited at the moment. Basically the most useful thing anyone can keep with them when on site is an illustrated map with the major buildings at a 2.5D angle, available printed out on pamphlets available from the Hugh Owen Library, Reception and other key visitor areas, or as a PDF available from the University's website. Whilst the map provides a starting point to identifying the rough location of key buildings, it does not provide much route guidance or, crucially, any real help to people of reduced mobility. The University does not have a smartphone app which people could use to help them navigate around campus like many other universities in the UK do.

It is a given that virtually all students these days have a smartphone on them at all times, so the University (specifically Student Support who are responsible for promoting accessibility) started an initiative called AccessAber, a mobile web app which can be used by any modern smartphone to assist people of reduced mobility around their campuses. The app uses static maps with hard-coded routes between buildings. This project was suggested to see how a native smartphone application can bring enhancements and a better user experience to users which the web app currently can not provide.

1.2. Analysis

I looked at the problems which the core audience for the app face and what technologies are available to native iPhone applications, many of which can not be used in a web app.

Here are some of the technologies available in native applications which could have been used in the project but were rejected:

Bluetooth beacons

Used to provide accurate positioning information where GPS may not be able to, specifically indoors. This would be good for providing accurate indoor routing, but the

problem of cost of deployment and maintenance of the beacons meant that this would be a non-starter for this project.

Augmented reality
Providing a 'heads-up' display of route guidance and information on top of the scene where the user is through their iPhone's camera. Conceptually a cool idea, but adds limited actual value to providing accurate and useful routes through campus, requires very precise GPS and digital compass positioning data and a lot of up front work in modelling the campus in 3D space.

3D maps
This would perhaps be the most beneficial feature to the application, as it would allow users to see more clearly the vertical relationships between buildings and paths on the steeply graded Penglais campus. Top-down 2D maps can not give a clear indication that, for example, the Llandinam Building is on a significantly different ground level than the Hugh Owen Building. The amount of work that would have to go into modelling the Penglais campus and its buildings in 3D space would far out way the amount of time which one person could spend on this project, and there is also the significant challenge of showing routes which go underneath and through buildings.

In the end it was decided to design and implement an app which can add real value to top-down 2D maps, providing data and context which is not currently available in established smartphone map apps like those from Google or Apple which will be useful to people of reduced mobility.

Research was taken into how the map should be designed and implemented, mainly what data could be used as a starting point. The amount of work required in providing data for the map should be kept to a minimum – basically, the project should not have to create a new map of Penglais campus from scratch as this takes a lot of time to get right.
The AccessAber web project uses bitmap tiles provided by OpenStreetMap, a community driven global mapping project where anyone can contribute to the dataset and which is free to use (within their terms of use, which mainly concerns the amount of bandwidth their bitmap tile servers can be used). The OSM dataset for the Penglais campus is already highly detailed, as it appears to have been contributed to by users at Aberystwyth University, and maps out all of the buildings and paths around Penglais campus, critical data which is missing in Google and Apple maps, which only show the roads and is not very helpful to students or visitors.
The OSM data is available as pre-rendered bitmap tiles and as XML data. The XML was a key advantage of using OSM, as it holds metadata for the elements of the map, like if a path has steps, and if a building entrance is wheelchair accessible. This metadata can be used by an app to know if a path is suitable for people of reduced mobility, information which can not be determined by just having bitmap tiles.

If the primary use case of the app could be distilled down into one sentence it would be "Get me to X". Users will mainly want to use the app to know where they are and what is the best route to get them to another place on campus. Visitors and new students may not know what building they need to go to for a particular department or service, and/or where that building is, so the app should also have a directory of buildings, departments and services, which allows users to search for and browse for the destination that they want to go to, rather than just selecting a position on a map.

During the analysis of the main paths through the campus it became obvious that a lot of them go through buildings. For example, the Llandinam building has a set of automatic double doors to get from the Llandinam concourse area to the Physics

building and other nearby buildings. The alternative of not using this route would be to go all the way around the Llandinam building on the main roads, making the path significantly longer. Indoor routes are not part of any map available for Penglais including the OpenStreetMap. It was decided that if the major indoor routes could be added to the map it would not only show these significant shortcuts but be of great benefit to wheelchair users.

Adding indoor routes presented a number of challenges, including how you would represent them on the map, how they could be differentiated from the normal outdoor paths, and how to handle the problem of having multiple building levels with their own paths on a flat 2D map. The OpenStreetMap project has looked at this problem and has presented some solutions, which some buildings in the global OSM dataset has used. The most active solution, of tagging ways with metadata which describes that they are indoor and what level they are on, was chosen to experiment with. [**1**]

1.3. Process

During the research and evaluation stage of the project several prototype iPhone applications were developed to evaluate the feasibility of using some of the technologies and frameworks available in the SDK, such as using Apple's Map Kit for the map part of the app and Core Data for the directory model. These were written with no formal upfront design, just using best design and implementation practices dictated from previous development experience with the system and Apple's developer documentation.
These "spikes" allowed me to discover the potential and limitations of such technologies, and helped to formalise the final requirements of the deliverable implementation. I wanted to keep the scope of the implementation to using as much of the existing frameworks available in the SDK as possible, and to not try to "reinvent the wheel" where a suitable, well proven technology can be used as a solution to the problem, even if it does not provide all of the features which I may have wanted in the final application. Since these frameworks are already well tested and supported, it reduced the need for low-level implementation details (like view drawing) and testing, and would allow me to focus on the parts of the app which actually add value, specifically end-user features.

For a time development work progressed on the spikes and final features were formalised. Some of the spikes were thrown away, while others continued to be developed until eventually they formed the main codebases. Development progressed on two apps, one with the map and OSM model, and another with the Core Data model to browse departments, buildings and services. Integrating the two apps into one was relatively simple as the two models are not dependent on each other. The Core Data directory model has references to the OSM model through unique identifiers in the OSM XML data. For the app to show a building on the map which the directory references, the building ID is simply looked up in the OSM model.

As a result of developing the app from spikes no formal up-front design existed for the app, and design was iterative. This can be viewed as an agile approach. An up-front design would have been inappropriate for this project as I was not familiar with the frameworks I would be using, and final features were not finalised until after experimentation and evaluation of the frameworks. Refactoring and best practices though using established iOS development patterns played a key part in keeping the overall architecture of the app usable and extensible.

2. Design

2.1. Overall Architecture

The app adopts traditional iOS Objective-C development design patterns.
As well as the obvious patterns which come with any object oriented language (Encapsulation etc.) the native application frameworks used in iOS development are designed with strongly defined patterns which most iOS applications also adopt in their own code to ease integration with the frameworks.

Model-View-Controller

The main design pattern which Apple uses with its GUI frameworks (including Cocoa Touch, the collection of classes for developing native iOS GUI applications). MVC dictates that code which defines the UI of the application should be in separate classes from code which manages the application data, with controller classes as a mediatory between the two ("views don't own data"). This is to encourage reusability and loose coupling, making a growing application easier to manage and maintain. In the AccessAber iPhone app, the layout of the views are laid out graphically in a Storyboard (the View), and then each one is connected up to it's own UIViewController subclass (the Controller), which controls that particular view's behaviour and mediates the communication between it and any Model classes which it needs to use to display and change data.
Other custom controller classes exist in the app which are not associated with any particular view. These include the Application singleton controller which owns and manages global application state.
The application has and manages two separate data models: a Core Data model which has data for the departments and services of the University, and an OpenStreetMap model which has the map data for the Penglais campus. Both of which are discussed later on. These models have their own classes for defining each type of object in the model (a.k.a. entity) and a model controller class which holds references to all loaded model objects and manages the state of the overall model in the application.

2.2. OpenStreetMap

OpenStreetMap (OSM) is a free, collaborative mapping project where anyone can contribute to maps anywhere in the world. As a result, it often provides must better accuracy and detail than other, closed mapping systems, like Google Maps or Apple Maps. People familiar with areas can contribute to the dataset, adding details like walkable paths, buildings, stairs, building entrances etc. The Penglais campus already has a lot of detail in OSM thanks to past and current contributors from Aberystwyth University, some of which may have been contributed for the AccessAber web project, which uses OSM map tiles. Furthermore, the raw XML data for the maps are available for anyone to download and use for any application, as well as pre-rendered map tiles.
The OSM project wiki defines a very simple XML schema for describing maps, and that schema was used as the basis for the map data model inside this app.

- Element: The base type for any OSM element (Nodes, Ways and Relations). Each element has an identifier (a 64-bit integer) which is unique in the whole world. Also, any element can have tags, key-value pairs which provide metadata about the element.
- Node: A point on the map with a latitude/longitude coordinate.
- Way: An ordered collection of nodes. Ways can either be open (where the collection of nodes describe a line, which can be a road or a path for

example) or closed (where the last node is the same as the first, describing a polygon, such as a building or area).
- Relation: A collection of nodes and ways. Used to group elements which are related to each other. For example, all nodes and ways within the Penglais campus are part of a relation with a name tag of “Aberystwyth University”.

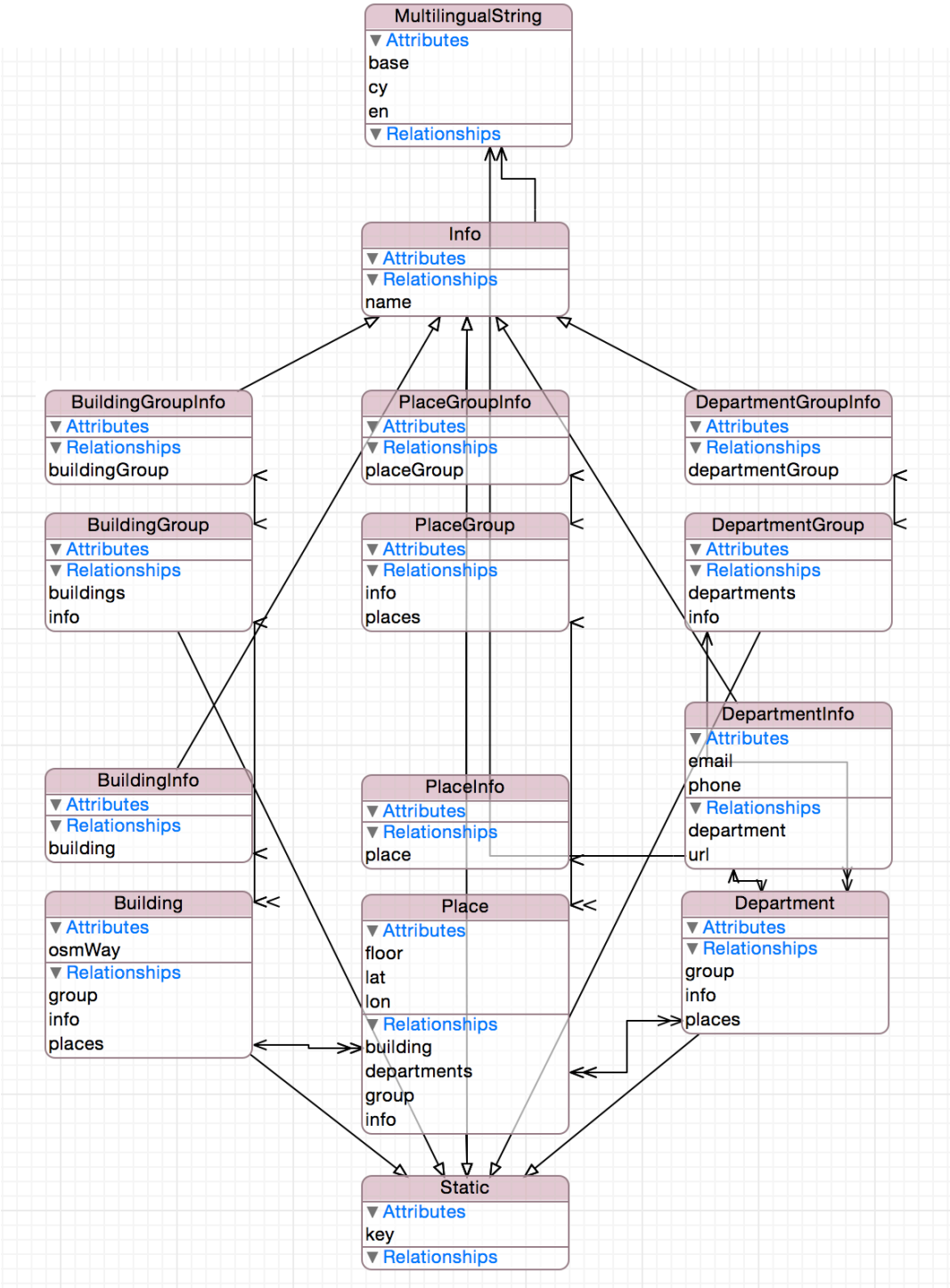
These data types were implemented in the app as part of the OSM model (classes with the OSM prefix).
The OSM XML data for the Penglais campus downloaded from the OSM servers is included as part of the application bundle. This XML data is parsed by the app on launch and the OSM objects created in a shared OSMModel object, which holds references to every OSM element read in. These objects are then used by the MapController to create the visible vector graphics used in the map view and by the route finding algorithm.
The XML can be edited graphically in a number of free tools. The most prominent one is JOSM (Java OpenStreetMap editor). This tool was used for the project to inspect what data is available for the Penglais campus and to add details which may be useful for the application. The main addition is indoor mapping. The main routes through the Llandinam Building were added. In the future, anyone using the JOSM tool can add indoor routes to other buildings and the app should be able to use them by updating the included .osm XML file included in the app bundle.

2.3. Core Data

As well as having maps, the app has a directory of University buildings, departments and services (e.g. libraries). This allows users to browse for locations which they may want to get to at a logical level. For example, a Computer Science student can go to the Computer Science department section in the app and get a list of all locations which Computer Science students may be interested in, including the Department of Computer Science itself in the Llandinam Building, the computer rooms such as the Delphinium, and major lecture rooms such as those in the Hugh Owen Building or Physical Sciences. Each department can also have other useful information shown like the department website and phone number.

At a logical level, there are relationships between the buildings, rooms and departments. Designing how these would be represented in the app took that into consideration and it was decided to use a relational data structure to store and manage the data. The iOS development platform provides a framework called Core Data, which is a relational object management and persistence framework, similar to EJB in Java. Entities can be defined and relationships established in a Core Data model, and then the framework manages the backend database for the developer.

The graphical representation which Xcode produces of the model:



One of the features which Student Support wanted in the app was to support English and Welsh. This presents a problem in the Core Data model, as the attributes which an entity can have is limited to basic data types, mainly Strings and Numbers. To support two languages, each attribute which displays information to the user, like building names, would have to support two values. To do this, a MultilingualString entity was created which has two String attributes, one for each language. Instead of having a String attribute in an entity to represent a name for example, instead a relationship was established to a MultilingualString entity object. The code interprets a MultilingualString entity relationship and selects the correct String depending on the user's device language setting.

3. Implementation

3.1. Route Finding Algorithm

A key functionality of the app is to find the best possible routes for users from one place to another. The web version of AccessAber used hand-coded routes between buildings. Whilst this approach would guarantee that a route would be appropriate, as it had to be decided by a human instead of a computer, it involves a lot of work on behalf of the developers and maintainers to cover all combinations of buildings. For this app I wanted the people maintaining the app to have to input as little data as possible, so I wanted the app to come up with appropriate routes by itself, and also for it to be able to produces routes from any location around campus, not just from buildings, by using the user's GPS location, which may be outside somewhere. For this a routing algorithm was required to produce any possible route which the user may want.

Research was taken into how other map applications, on phones and on websites, implement route finding. It makes sense that the routes be the shortest possible along appropriate paths which the user can take, so shortest route algorithms were researched.
Virtually all of these algorithms use graph theory to represent the data structure of the map, and then to find the shortest possible path though the graph from the source node (the users current location) to the target node (where they want to get to).
The OSM data model closely resembles a graph. It has nodes which are locations in the world, and ways which are connected nodes which can be seen as graph edges. However, ways do not properly resemble graph edges, as nodes can be shared between multiple ways where paths can diverge. A way does not start/end at every possible turning point in the map. So edges need to be created for a graph algorithm in which every node is connected to all nodes which a user can move to.

After researching what algorithms other route finding solutions using OSM data use I came across the Dijkstra algorithm, which is a simple but effective algorithm for finding shortest paths between nodes along edges which have distance values (also known as weight in general graph theory).

The implementation of the algorithm in the app is modified slightly in several ways: First of all, the search space of nodes and connections between nodes is limited to only those which the app deems to be navigable for the user. Only nodes from ways tagged "highway" are used, which excludes all nodes and ways which describe buildings, green areas etc. If the user has selected an option in the apps preferences which says that they cannot take stairs, all ways which are tagged with "steps" are also excluded. Finally, any paths which are tagged with "dirt" are also excluded as these tend to be shortcuts through woods which would be inappropriate.

A graph is built up as the algorithm is running. The app gives the algorithm a source node and a target node. From the source node, the distance of all adjacent nodes in each navigable way that the node is part of is calculated by the Core Location framework using the nodes' lat/lon coordinates, and are placed in a priority queue, where the node closest to the source node is placed first. This process repeats for each node from the start of the queue until eventually the target node is reached. The node distances are accumulative, so the calculated distance of an adjacent node is the previously calculated distance of the current node plus the distance from that node to the adjacent node. The algorithm keeps track of previously visited nodes so

that it does not visit them again and each node has a previous node to keep track of the path used to the node.
When the target node is reached, the path is walked back using the recorded previous nodes. This should always yield the shortest path from the source node to the target node.

Here is a simplified pseudo-code version of the algorithm:

```
allHighwayNodes = (all navigatable highway nodes)
nodePriorityQueue = [sourceNode]
nodeDistances{node:distance} = {sourceNodeId:0}
previousNodes{node:node} = {}

while (nodePriorityQueue.count > 0) {
	currentNode = nodePriorityQueue.pop()
	if (currentNode == targetNode) {
		success = true
		break
	}
	adjacentNodes = getAdjacentNodes(currentNode)
	foreach (adjacentNode in adjacentNodes) {
		dist = nodeDistances[currentNode] +
			distanceBetweenNodes(currentNode, adjacentNode)
		if (dist < nodeDistances[adjacentNode]) {
			nodeDistances[adjacentNode] = dist
			previousNodes[adjacentNode] = currentNode
			nodePriorityQueue += {dist:adjacentNode}
		}
	}
}

if (success) {
	routeNodes[]
	currentNode = targetNode
	while (currentNode) {
		routeNodes += currentNode
		currentNode = previousNodes[currentNode]
	}
	return routeNodes
}
```

3.2. Map

The map part of the app shows the user all of the buildings and paths throughout Penglais campus. It relies on the highly detailed OpenStreetMap data already available for that area, where Google and Apple maps only show the roads with no buildings. The AccessAber web app also uses OpenStreetMap data, but it uses the pre-rendered bitmap tiles which OSM provides on their tile servers. While the bitmap tiles show the detail which is in the OSM dataset, it is very limited in how it can be used. First, the app can not configure what is displayed in the bitmap tiles or how the various elements of the map are drawn. This results in a poor visual representation when you try to add your own custom elements to the map, such as routes. For example the bitmaps have the names of the buildings and areas on them, and they can not be hidden or made bigger/smaller. Also the quality and resolution of the bitmaps are limited especially at the high zoom levels which a walking/wheelchair navigation app requires. Most importantly however, the bitmap tiles do not provide any actual data to the application as to what they are showing. The app needs to know the coordinates of places and paths to be able to provide routes and show points of interest such as lifts.
Luckily, the raw XML OSM data is available to download as well. The OpenStreetMap dataset is represented in XML form and a subset of the global dataset can be downloaded for a particular area. This data is what the bitmap tiles are rendered from, and can be used by any other renderer to produce custom bitmap

tiles. And the data contains all of the coordinates and metadata associated with each map element which the app can use to produce routes. The format of the OSM XML data is described in the OpenStreetMap section of this report.

It was decided to use the raw XML data to produce map logic and map graphics. It gives the app the flexibility to show only what is relevant to the use case of the app. There were two possible routes to producing map graphics from the XML data: Pre-render custom map tiles, or render the map in real time as vector graphics. The latter option was chosen because it removed the need to develop a tool or script to render tiles across the campus at all zoom levels, which would significantly increase the download size of the app and result in developing a separate app to produce the tiles, and it allowed the app the flexibility to render any elements of the map it wanted to depending on the current state of the application in real time. Finally, rendering a vector map instead of bitmaps made the map look clean and easier to look at.

The main problem of rendering map graphics in real time is how it will effect performance. The map contains a lot of elements and it was initially worried that trying to render all of the buildings and paths as vector graphics as the user is panning and zooming the app would result in unacceptable performance and memory issues.

A spike app was written to test how an app would perform trying to render all OSM elements on a map in real time. MapKit, the Apple framework for including Apple's Maps in 3rd party apps, has APIs for adding overlays on top of their own maps. The overlays can be polygons, lines or other shapes and the drawing of them are fully customisable. This was a great thing to discover as it eliminated the need to write a custom renderer, which would take a significant amount of time. When each OSM Way (collection of nodes forming a shape) were added to a MapKit view, performance was surprisingly good. Apple have clearly put a lot of work into optimising their MapKit framework and do a lot of background rendering and caching to keep performance high and memory utilisation low.

This approach also allowed indoor maps to be shown, which were not at all visible on the standard OSM pre-rendered map tiles. MapKit has APIs for mapping elements to geospatial coordinates, making the connection between graphics and real-world spatial data easy. With such information, the app can know when the user has zoomed into a particular building (when a building overlay consumes most of the screen using centre coordinates and zoom level) and can then render the indoor ways associated with that building through OSM relations.

4. Critical Evaluation

While I believe the set of features that I wanted to include in the app were a good start at tackling the problem of helping people of reduced mobility around campus and that the design and architecture of the app were more or less solid, issues prevented me from completing the implementation and providing a detailed report. Clearly a lack of testing and documentation (code comments or manual) are a major failing of the development side, though I do not regret not using a more formal development strategy. A lack of project planning and personal issues beyond my control contributed to a major failing of this project.
It is hard to say what I would have changed or done differently in respect to the finished product as it was not completed. Indoor mapping was seen as a valuable new feature over any other existing solution, but requires the developer or maintainers to painstakingly map out indoor routes, and despite a draft specification provided by the OpenStreetMap wiki for doing indoor mapping, there is no tool support in JOSM to make this job easy.
While the map provides the best possible view of the Penglais campus the complex nature of buildings and paths still make it difficult to make sense of the area from a top down 2D map, so perhaps some more research and experimentation could be taken into less conventional mapping to try and improve the user experience.

5. Appendices

No third party code was used in the application. The frameworks and signing of the app to enable it to run on iPhones are bound to Apple's Developer licence and terms.

JOSM was used to download and extend the OSM dataset used by the app. This application is licenced under the GPL:
https://josm.openstreetmap.de

6. Bibliography

[1] OpenStreetMap wiki. OpenStreetMap wiki. [Online].
http://wiki.openstreetmap.org/wiki/Indoor_Mapping