Cross Platform Application using Ionic Framework

Darren Flannery and Peter McCaffrey

BSc in Computing in Software Development
Galway Mayo Technical Institute
http://www.gmit.ie

Abstract. This dissertation examines the advantages and limitations of cross-platform mobile application development by building a smartphone application using the Ionic framework. We created an application for the use of Android, Iphone and Windows smarthphones. The implementation of the application was achieved with a single codebase of the standard web development technologies - HTML5, CSS and JavaScript. During the process we reviewed other frameworks and technologies, some of which assisted us with our development. This report will reveal how we implemented the application and problems we encountered during the process. Our result was a fully functional platform-free and responsive application.

1 Introduction

Within the past few years there has been a significant increase in the sale of smartphones and tablets. People are rapidly downsizing from desktops and notebooks to mobile devices. The term "mobile device" has been traditionally used to refer to small hand-held devices such as mobile phones with limited screen size and processing power. However, A new market for smartphones and tablets has emerged since the release of the IPhone and IPad, giving a new meaning to the term "mobile device". The IPhone was the first large touch-screen smartphone to achieve commercial success. Since then, the sale of mobile devices vastly accelerated. According to a research report [13], it is expected that by 2017 the sale of tablets alone will nearly double the sale of PC's (Desktop and Notebook).

As this growth continues, the number of mobile apps being downloaded is rising drastically. Billions of apps are being downloaded every year on a range of different Operating Systems, and on all device sizes, from smartphones to tablets.

Mobile applications are currently at a stage where they have to exist on multiple platforms, and cater for all device sizes in order to reach their full potential audience. This causes concern for developers as each operating system is tied to its own programming language. In order to build applications for several platforms, the developer must build several different applications, one per platform. This extra work results in greater expenses.

The arrival of HTML5 and it's growing support in mobile browsers has meant that web applications have become a popular alternative to native application development. However, web applications do have limitations, but because of the increasing popularity of this approach, there are technologies such as Ionic Framework that allow developers to use web code to produce cross-platform applications.

1.1 General Problem Statement

We intend to develop an application for all mobile devices by using Responsive Design and Cross Platform Development with the Ionic Framework. We will also investigate the advantages and limitations of web applications, Responsive Design and Cross-Platform Development.

1.2 Objectives/Aims/Hypothesis

We were given the task of developing a mobile application for a client that could be used as a self-guided walking tour. The name of the walking tour is "Galway City Waterway Walks". The client intends for the application to be a mobile-app version of a self-guided tour book, but with an interactive map and a direction service.

The app will use geolocation to get the users current location. Using the users location, the app will then guide the user from start to finish, as well as providing information on famous landmarks along the trail. The client intends for the application to be able to run on any device so as not to alienate potential users.

Our main objective is to develop a responsive, cross-platform application which uses GPS tracking to guide users through walking trails of Galway city. With this assignment, we hope to become more competent with cross-platform mobile application development.

2 Method

2.1 Web Application or Native Application

Our fist thought was to design a web application over a native application. This will ensure that it can run on any device as all smartphones carry a web browser. A concern for this approach however, was how the application would appear on smaller devices. We considered following the principles of Responsive Design to ensure the application would work on smaller devices. Our main concern with Responsive Design was it does not incorporate all the smartphone's features like the camera or GPS, unlike a native mobile-app. Another consideration was the speed of the application; A native mobile app will provide users with unique functionality and speed that can not be achieved with a responsive web application.

However, with development tools like Cordova/Phonegap and Ionic available, it would make it possible to design a web application in HTML5, CSS and Javascript and then package the application into a native mobile application for multiple platforms. This approach would also give the application access to the smartphone API's we needed such as GPS.

2.2 Cross Platform Development

Cross-Platform Development is the process of writing mobile applications for multiple operating systems.

Cross-Platform application is a complicated process and there are several issues that must be overcome in order to develop applications for multiple platforms. The first and most obvious obstacle is the language with which the applications are written. Other obstacles include differences in hardware for different devices. Also, each platform may have a set of individual requirements and guidelines with which the developer must follow.

However, with certain developer-tools it is possible to use the same source code when developing for multiple platforms. The application is written using HTML5, CSS and Javascript, which the smartphone will then implement.

2.3 Responsive Design

Responsive Design is the practice of designing a website or application that responds to the size of the window or device with which it is being viewed. The goal of Responsive Design is to create one website or application that can be viewed on any size of device while minimizing any loss of functionality or readability.

Responsive Design using the following tools to achieve resolution independence:

- **Flexible Grids:** Page grids that scale with screen resolution instead of using fixed pixel dimensions.
- **Flexible Images:** Images that scale to the size of their surrounding grid.
- **CSS Media Queries:** CSS styling tailored to ranges of resolutions or types of device.

The benefit of responsive design is that it offers cross-platform use and is resolution independent. It also works well on most browsers. However, as we reviewed earlier, it has it's limitations. It does not harness the full potential that a native application has over a mobile device.

3 Literature/Technology Review

Responsive design was the first step to creating platform-free applications and websites when the term was first coined in 2010. This literature review will look at the use of Responsive Design and it's limitations. It will also reviews other alternatives approaches to creating a platform-free mobile presence. We will then examine how hybrid applications favour over native applications.

3.1 Traditional Design Methods

Before the introduction of Responsive Design, developers and designers followed the principles of "pixel-perfect" web design. The method for creating a pixel-perfect website is to design a mock-up in Photoshop and then to recreate that design to fit the browser, like you would for a printed piece of paper. Kim [9] points out that this design is flawed as unlike paper, web browsers are dynamic. The user has the ability to resize the browser which can cause the design to break.

From the the beginning of the World Wide Web up until the 2000's, the majority of screens resolutions were 800x600 or 1024x768. This made wire-framing and design a lot simpler compared now, with the selection of devices on the market today [20]. With the rapid growth of smartphones and tablets, web pages are being viewed on a variety of resolutions like never before.

Although it is possible to view these designs on smartphones, it is not an ideal solution. In a study done by Shrestha [19], the limitations of using a mobile phone on standard websites were examined. 75% of participants failed to perform complex tasks. In Joly's [8] opinion, institutions that don't "adopt a multi-device web strategy will definitely alienate web visitors who don't use recommended devices".

3.2 Responsive Design Method

The term Responsive Web Design (RWD) was coined by Ethan Marcotte [11] in 2010. The term was derived from a discipline called "responsive architecture", which looks at "how physical spaces can respond to the presence of people passing through them... architects are experimenting with art installations and wall structures that bend, flex, and expand as crowds approach them".

Marcotte applied this principle to web development, designing flexible layouts that respond to the "media that renders them". The main goal of Responsive Design is to create a flexible web page that will adapt to fit any screen size and that will look equally good regardless of the device [9].

Wisiniewski [22] describes responsiveness as a set of tools that assist developers in a "creating a single website that responds to context". In a responsive website, the browser can be resized without causing the design to break or causing any elements to be hidden from view. Responsive Web Design is seen as a long-term solution[4] as the market for web browsing devices is ever changing. It is now possible to create a single website suitable for all current, and future devices.

3.2.1 Considerations

As well as creating a layout that can resize itself, a good responsive website should also have the ability to rearrange , display or not display certain content and dynamically turn a top navigation menu into a drop-down menu [22]. RWD's main purpose is to support variations in screen sizes resolutions, web browsers and input [13].

Bin [24] and Pastore [17] also note that human behaviour should be taken into account in Responsive Web Design. For instance, the time spent browsing on a smartphone is usually below 15 minutes while the time spent browsing on a tablet or desktop is considerably longer.

Nebeling et al. [12] indicate that the principles of RWD should not just be applied to smaller mobile devices, but larger ones also. There is a growing trend towards large high-resolution desk monitors. The term "desktop" may now refer to a wide range of displays and web developers should take this into consideration when designing a responsive site.

According to Snell [20], creating a responsive site requires much more time and effort than a traditional "pixel perfect" site. It is not just creating a desktop site with a mobile afterthought, it requires coding multiple websites at once.

3.2.2 Performance

A study completed by Guy Podjamy [2] revealed, that when 347 responsive websites were tested on various screen resolutions, 86% of the subject sites "weighed" virtually the same on all sizes. This means that it is common for developers to not take performance into consideration when designing a responsive website.

Wisinewski [22], Bin [24] and Fox [7] acknowledge that RWD is more than creating flexible layouts, but controlling the media that is sent to the device depending on it's bandwidth and screen resolution. In terms of bandwidth, Wisinewski [22] states it would be more optimal to serve a smartphone user a smaller version of an image than what would be sent to a PC user. Wisinewski [22] and Fox [7] also point out that high resolution images should only be sent to devices with a more expansive viewport.

3.3 Implementing Responsive Design

Responsiveness is achieved by a blend of web technologies, design and coding techniques to achieve flexibility for wide or narrow devices. The traditional fixed-width layouts are replaced with a more adaptable fluid layouts, by using maths and percentages, or setting hard values and then hiding certain portions of the element. However, Snell [20] suggests completely abandoning pixels in favour of percentages when notating sizes in CSS.

3.3.1 Approaches

According to Mohorovicic [13], there are two approaches to for the implementation of Responsive Web Design, "Mobile first" and "desktop first". In mobile first, a layout is defined for smaller screens and then it is progressively enhanced as the resolution increases. Desktop first takes the desktop resolution as a the starting point, and then designs accordingly as the resolution decreases. Snell [20] and Wisniewski[22] both dis-encourage starting from a desktop design and recommend a mobile first approach. Snell [20] advises to begin with how the elements behave below 1000px and design from there.

3.3.2 Viewport

The first step for implementing a Responsive Design is to define the Viewport. The viewport is the part of the page currently shown on the screen. It has become a standard for smartphones to zoom out to display the full width of a non-responsive website on the screen. Setting the viewport to the width of the device will prevent this and allows you to design to based on a specific resolution [17] [20].

3.3.3 Features

Marcotte [11], Kim [9], Natda [15] and Mohorovicic [13] state that there are three main features of Responsive Design. The three main features that they highlight are: Fluid Grid, Flexible Images and Media Queries.

Fluid Grid layouts replace the traditional Fixed Grid layouts. Fixed Grids are contructed by specific pixel measurements while Fluid Grids use percentages. This method allows the grid to resize itself depending on the size of the browser. "You can adopt the structure of your site for desktop screens, for tablets and smaller mobile devices." [15].

Flexible Images follow the same concept as Fluid Grids, substituting pixels for percentages to so the images move and scale proportionally as their container resizes. Bin [24] noted that by using the "max-width" property for images, he was able to constrain the size of the image, while allowing it to adapt to smaller devices.

Media Queries were intorduced by W3C [23] as part of the CSS3 specification. Media Queries are "conditional statements" that can identify the media type, and characteristics of the device and browser. With media queries we can serve different Style Sheets depending on the width of the browser [13]. Media Queries are not only useful for resizing, but also for rearranging. Elements on the page can be reorientated to stack if the screen is a certain pixel dimension [7].

A breakpoint is the width of the browser when media queries are applied [15]. Ethan Marcotte [11] lists standard breakpoints to use when designing a responsive site based on device sizes (e.g. 320px, 600px), however Wisniewski [22] argues that it is better for the developer to determine their own breakpoints by testing each design at different screen sizes on an individual basis.

3.4 Other Approaches to Mobile Optimization

While most are in favour of RWD and believe it to be the best solution to mobile optimization, some argue against the idea. Dysart [3] claims that bad design can neglect larger screens causing large text and overblown features, and that responsive websites are more trouble than they are worth.

Mohorovicic [13] points out some problems with RWD. He notes that some browsers are not compatible with CSS media queries. He also states that performance is the biggest problem, mainly because of large, unnecessary images being served.

Responsive Web Design is not the only solution to creating a mobile friendly online presence, below are some other approaches.

3.4.1 Mobile Site (M Dot Com)

As mobile browsing has exploded in the last several years, companies started to adapt to this the mobile web before RWD. The term "mobile site" refers to a septate website designed for mobile devices. The site would detect when you are using a mobile device and redirect you to an alternatite website, with a different URL. Usually something like: "m.websitename.com".

Serrano et al. [18] point out the advantages of choosing a mobile site over a responsive one: "It usually has a better feel than responsive web because it renders the user interface controls in a way that's similar to a native app"

Fox [7] shows favour to responsive sites over "mobile sites", claiming that responsiveness allows focus to be shifted towards content rather than devices. He states that with a single web presence there will be continuity from one environment to the other and will avoid confusion to the user.

Joly [8] also compares "mobile sites" with Responsive Web Design but also shows favour to RWD, stating "it is more practical for a single set of code to serve all connected devices".

Many argue against mobile sites for the reason that it will require designing another version of your already existing site [17] [4] [21] [7]. However Nebeling et al. [12] point out that responsiveness is also difficult to apply to existing sites and requires major re-engineering.

3.4.2 Native Apps

Since mobile devices adopted web-based operating systems working in the perspective of "always-on", another approach towards a mobile web presence has been the Native Application [17]. Although Pastore [17] recommends a responsive website, she does point out advantages of developing a Native App: "Potentially lower network usage at runtime" and "access to platform API's" (Camera, GPS, etc.).

Pastor [17] and Thacher [21] point out that there are major disadvantages of Native Applications compared to Responsive Web Design. They note that Native App's are tied to their proprietary software, meaning several versions (Android, IOS, etc.) would be required on top of a website or you risk losing a large percentage of your audience.

Florins et al. [6] argue against specific interfaces for each platform, stating that it does not guarantee consistency between the two designs.

3.4.3 Hybrid Apps

Android applications are written in Java using an API designed for mobile applications. Other Smartphone platforms, such as Apple's iOS or Microsoft's Windows Phone 7, differ greatly in their native application programming model. These three platforms are not compatible to develop across.

Android primarily uses java but you can develop with C and C++, while windows use XAML and C#. IOS on the other hand uses xCode, C, and C++. All of these platforms can however run JavaScript and HTML meaning that web applications are compatible with each device.

There are programs which can make these platforms compatible such as Phonegap, Ionic Framework, and JQuery. Phonegap is used to convert the HTML website to a mobile platform and package every component so that the application will be compatible with platform operating system and phones. As there are hundreds of different variations of phones, many of which use different operating systems and sensor, technology like Phonegap has become an increasingly important tool in mobile development. Frameworks such as Bootstrap are used to re-size web pages making them fit to any screen and giving them a responsive elements such as text, menus, images and many others. Unfortunately this only works for web pages and not native phone applications. However since each of the major phone platforms are capable of running JavaScript and HTML, frameworks such as Ionic can be used for creating hybrid applications.

3.5 Literature/Technology Review Conclusion

This literature review reveals a move away from traditional web design towards Responsive Design. Responsiveness is not only a way of adapting to today's technology, but it is a way of "future-proofing your website" [22] for technology that is yet to come. It is a device-independent solution, that reduces cost and maintenance.

This shift towards Responsive Design shows that it is no longer adequate to create a website or application with one device in mind.

Responsive design is becoming a standard for present web design. It will continue to evolve and looks to have a promising future.

4 System Architecture/Software Design

4.1 Deciding on a framework

After reviewing our options, we decided to build a responsive web application using the standard web technologies (HTML5, CSS, JavaScript). We will then package the source code into native applications for the three most used platforms - IOS, Android and Windows.

Before we decided to use Ionic we looked at available frameworks that could assist us with this process:

4.1.1 JQuery Mobile

jQuery Mobile is a cross platform mobile framework designed to simplify and enhance the development of mobile web applications by integrating HTML5, CSS3, jQuery and jQuery UI into one framework. It is similar to Ionic, that it is used to ease the development process and it comes with pre-defined themes and icons. However, after doing some research, we found that it has become out-dated and has poor performance compared to Ionic. JQuery Mobile is more focused on Mobile Web Applications, while Ionic was built for developing Hybrid Mobile Applications.

4.1.2 Phonegap/Cordova

Cordova is a platform for building native mobile applications using HTML, CSS and JavaScript. It allows developers to use web technologies to build apps for various platforms. The idea is to build one singular web application based on web technologies and Cordova takes responsibility of building native applications for each platform. It also gives JavaScript access to plugins such as battery status, motion sensors, file transfers, geo-locations etc. Cordova builds these JavaScript plugins every time the application is deployed.

4.1.3 Ionic and AngularJS

Ionic is a front-end framework for creating mobile apps. It is free and open source, offering a library of HTML, CSS and JavaScript components and tools for building hybrid apps. Ionic is built on Cordova, AngularJS and SASS.
Ionic is similar to Twitter Bootstrap, as it is a collection of tools. It is used to ease the development process of building Cordova/Phonegap applications. It comes with a collection of built in CSS classes, icons, HTML templates and JavaScript libraries. Ionic makes developing with Cordova much easier.

– **Pros:** Faster development, working in web based technologies, which means less complexity. Building a singular app that will work on all platforms. Clean, simple, and functional. Ionic has been designed to work and display on all current mobile devices. With tons of popular mobile components, typography and interactive paradigms.
– **Cons:** Less performant than native apps on less capable devices, for example using Ionic application on early Android device, it will have complications. Limited to UI capabilities of modern web, for example, JavaScript or CSS may not have some of the same capabilities as programming languages such as Java or C#. If you were developing for Android, you cannot use java like you would for a normal Android application.

We compared Ionic applications to those made from JQuery Mobile and we felt that Ionic applications had a much nicer front-end. We were also keen to learn about AngularJS as it is becoming a favoured alternative to JavaScript and JQuery. We decided Ionic would be the best framework to use because of its recent popularity and the fantastic support it's website offers. More than any other framework, it is made just for developing hybrid applications.

4.2 Implementation

This section will cover how the application was constructed and how we utilized cross-platform development using the Ionic Framework.

An Ionic application is written in HTML5. Ionic uses Cordova to bridge the gap between the native language of an operating system and HTML by providing a wrapper for the app. Cordova then generates Javascripts used in the app for accessing the native API.

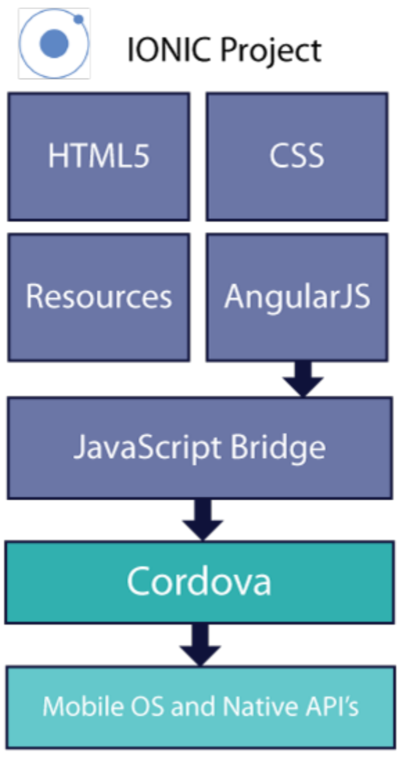


Fig. 1. Structure of Ionic Framework Project

4.2.1 Creating a New Project

To start the Ionic project, we fisrt installed Cordova and Ionic on the desktop. To create a new project is as simple as running the following code from the command line:

```
ionic start myIonicApp
```

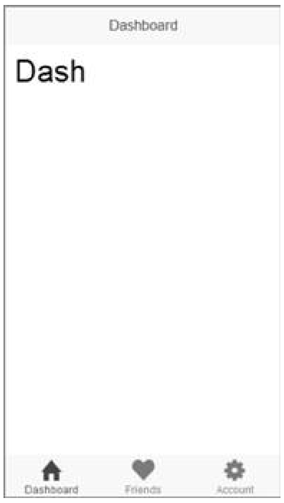


Fig. 2. New Blank Ionic Project

Once a project is created, the application is ready to be packaged into a platform application. This can be achieved for Android by running the following from the command line:

```
ionic build android
```

This creates an an Android Application Package that is ready to be installed on an Android device.

4.2.2 Project Structure

This section is an overview of the basic structure of our Ionic Application.

– **Views (Templates):** Stored in the /templates folder where each view is in a separate .html fiile. In our project we have five views/templates, one for each page and one for the tab menu.
– **Controllers:** Stored in /js/controllers.js. This is where the functionality behind the app comes from. It also controls the passing of data from the data layer to the view layer via data binding. Each view has its own controller. For example, the map and direction services are controlled by MapCtrl in controllers.js.

- **Services (Data):** The services.js file provides an Ionic app with data. We used it to create a datastore to provide the controller with longitude and latitude of Galway City for our weather feature on the Home Page.
- **App Configuration:** Stored in /js/app.js. This is where the states of the app are specified. The following code links the map view with it's controller:

```
.state('tab.maps', {
   url: '/maps',
   views: {
     'tab-maps': {
         templateUrl: 'templates/tab-maps.html',
         controller: 'MapCtrl'
      }
   }
 })
```

4.2.3 Setting Up The Map

The main feature of the application is the Map. The location of the map was specified in the tabs-map view. We used Google Maps API to add a basic map to our project.

```
<map on-create="mapCreated(map)"></map>
```

The functionality for the map comes from MapCtrl in controllers.js. We specified the longitude and latitude of Galway City and set up the Map Options and assigned it to the variable "map", which is then sent to the map view.

```
var galway = new google.maps.LatLng(53.273361, -9.057331);
var mapOptions = {
   center: galway,
   zoom: 15
};

map = new google.maps.Map($element[0], mapOptions);
```

The biggest challenge was specifying each walking trail on the map, and providing directions from start to finish. This was achieved by creating a DirectionsService object which is required to communicate with the Google Maps API Direction Service.

```
var directionsService = new google.maps.DirectionsService({});
```

We specified the "waypoints" along each trail, and then used a Directions renderer to render the direction results. This is shown on the map with a polyline.

```
directionsDisplayRoute1= new google.maps.DirectionsRenderer({
   polylineOptions: {
      strokeColor: "#00cc00",
      strokeWeight: 6,
   }
});
```

Another challenge was getting the users location via GPS. To achieve this, we ran the getCurrentPosition() method, if the getCurrentPosition() method is successful, it returns a coordinates object to the function specified in the parameter (pos).

```
 navigator.geolocation.getCurrentPosition(function (pos) {
   console.log('Got pos', pos);
   myLocation = new google.maps.LatLng(pos.coords.latitude, pos.coords.longitude);
   hereMarker.setPosition(myLocation);
   ...
```

We set the application call this method every 10 seconds to automatically update the users location.

```
//refresh every 10 seconds
setInterval(function(){
   myLocation = new google.maps.LatLng(pos.coords.latitude, pos.coords.longitude);
   hereMarker.setPosition(myLocation);
},10000);
```

Once we had the coordinates for the users location, we repeated the process of creating the direction service and the directions renderer to show a route from the users location to the start of the walking trail.

The landmarks were set up by creating an array of locations and giving the coordinates of each landmark.

```
/*INFO MARKERS*/
infoMarkers[0] = new google.maps.LatLng(53.269774, -9.053511);
infoMarkers[1] = new google.maps.LatLng(53.270152, -9.055572);
...
```

Then we assigned these locations to an arry of markers:

```
//create number of markers based on infoMarkers.length
for(var i=0; i<infoMarkers.length; i++){
   pointMarker[i] = new google.maps.Marker({
      icon: 'img/infoicon1.png',
      title: markerTitles[i],
      position: infoMarkers[i]
});
```

The building of the application was very challenging but was made easier with the help of the Ionic Framework.

4.3 User Interface Design

The application is constructed with different views that the user can navigate through. We wanted to keep the design of the application relatively simple to make it easy for the user to navigate through the application.

4.3.1 Navigation Ionic provides a range of CSS components such as Headers, Buttons, Lists, Tabs and Cards which allowed us to create a nice clean layout. We decided to use a tab menu that sits on the bottom of the screen and can be seen from every page on the app. This makes navigation very simple as it allows the user to see where they are and where they can go. This can be see in Fig. 3.

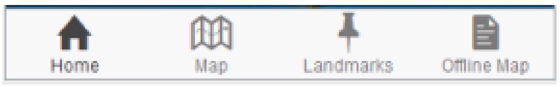


Fig. 3. Tab navigation visible from every page

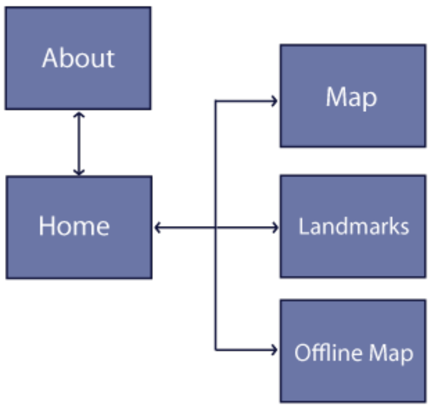
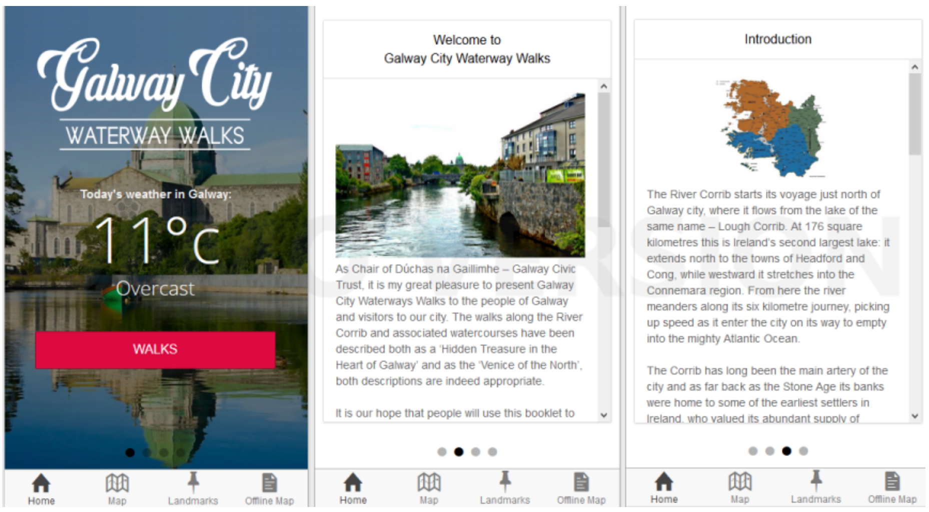


Fig. 4. Overview of the navigation through the application

4.3.2 Home Page We wanted to keep the "Home" page simple and modern. We used a full-sized background image which is a current trend for mobile applications. On the homepage we incorporated the weather for Galway City. This feature gives the user the current temperature and a brief written description of the weather. On this page, the user can swipe the screen to the left as the whole page is a slider. There are four full page slides, the "Home" slide, "About" slide, "Introduction" slide, and "Acknowledgements" slide. We felt it was important to incorporate these pages from the guide book into the application. We did not want to clutter the main navigation, so we felt this was an appropriate location.



As Chair of Dúchas na Gaillimhe – Galway Civic Trust, it is my great pleasure to present Galway City Waterways Walks to the people of Galway and visitors to our city. The walks along the River Corrib and associated watercourses have been described both as a 'Hidden Treasure in the Heart of Galway' and as the 'Venice of the North', both descriptions are indeed appropriate.

It is our hope that people will use this booklet to

The River Corrib starts its voyage just north of Galway city, where it flows from the lake of the same name – Lough Corrib. At 176 square kilometres this is Ireland's second largest lake: it extends north to the towns of Headford and Cong, while westward it stretches into the Connemara region. From here the river meanders along its six kilometre journey, picking up speed as it enter the city on its way to empty into the mighty Atlantic Ocean.

The Corrib has long been the main artery of the city and as far back as the Stone Age its banks were home to some of the earliest settlers in Ireland, who valued its abundant supply of

Fig. 5. Home Page with full page slider

4.3.3 Map Page The main feature of the application was the "Map" page. The guide book that we were working from uses a different colour for each walking trail - green, red and blue. We decided to incorporate this by using one of these colours for all buttons, icons and map polylines when the user was viewing a specific walk. For example, when the user was viewing Walk 1, all the features would be green and when viewing Walk 2, all the features would be red. This makes it very easy to distinguish each walk on the map and helps the user know where they are (Fig. 6).

The user can select which walk they want to view from a tab menu directly under the map. When a tab is selected, the chosen walking trail will appear on the map and the colour of the features on the page will change. This allows the user to change walking trail without having to navigate away for the current view.

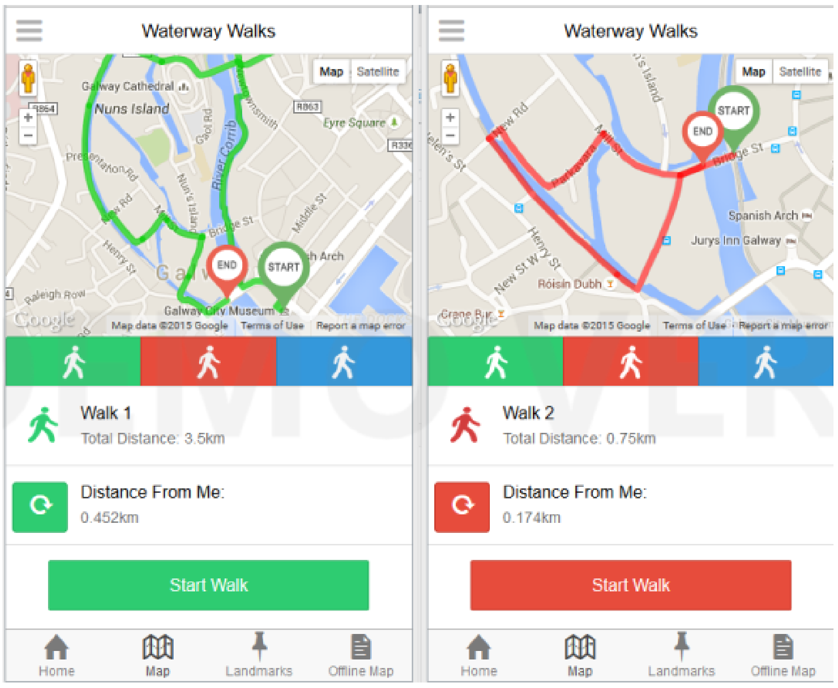


Fig. 6. Walk 1 (Left) uses green for buttons, icons and polylines and Walk 2 (Right) uses red.

On the top right hand side of the page, we added a "side menu" (Fig. 7), which slides on to the screen when tapped. The side menu contains:

– **Show My Location:** Toggle-button, which shows/hides a marker which represents the users location on the map.
– **Show Directions to Start:** Toggle-button, which shows/hides a direction service which guides the user from their location to the starting point of the current walk.
– **Show Landmarks:** Toggle-button, which shows/hides markers which represent the locations of Galway's famous landmarks.

The Map page also consists of a feature that shows your current distance from the starting point of each walk. This can also be seen in Fig. 6.

When a walk is selected, a "Start Walk" button appears at the bottom of the page. When this button is pressed, a pop-up window appears (Fig. 8). This window contains the walking-guide text for this walk taken from the guide book, along with images of each location. There is also an audio box in this window so the application can read out the text to the user while following the walking trail. There is a "Back To Map" button to close the window and go back to the map.

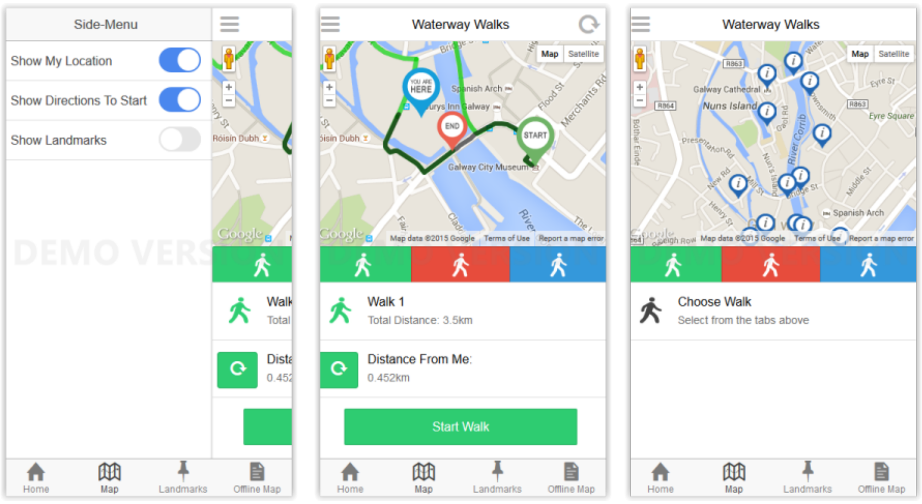
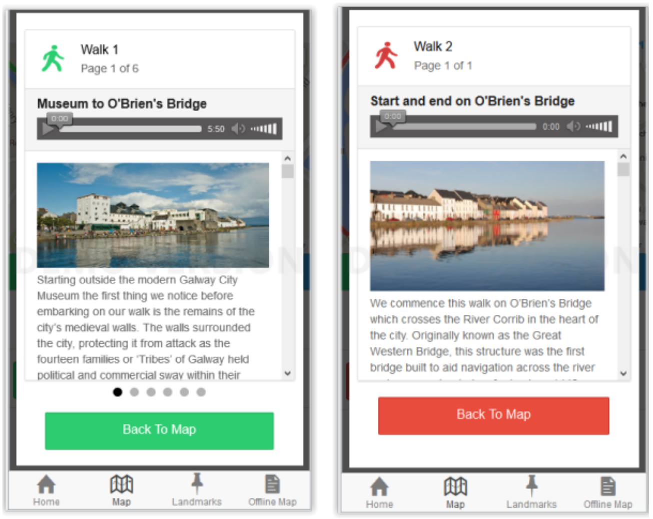


Fig. 7. Side-Menu(Left), Direction Service(Middle) and Show Landmarks(Right)

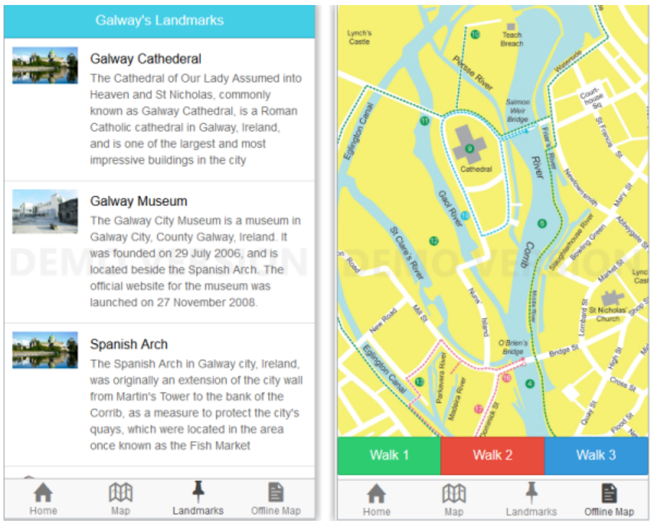


Starting outside the modern Galway City Museum the first thing we notice before embarking on our walk is the remains of the city's medieval walls. The walls surrounded the city, protecting it from attack as the fourteen families or 'Tribes' of Galway held political and commercial sway within their

We commence this walk on O'Brien's Bridge which crosses the River Corrib in the heart of the city. Originally known as the Great Western Bridge, this structure was the first bridge built to aid navigation across the river

Fig. 8. Pop-Up Windows, Walk 1(Left) and Walk 2(Right)

4.3.4 Landmarks Page and Offline Map Page We also included in the application a "Landmarks" page and a "Offline Map" page (Fig. 8). The "Landmarks" page displays a list of the local landmarks around Galway City and a short description of each one. The "Offline Map" page shows a full size JPEG image of the map and allows the user to use map with out the need to be connected to the internet. However, this page cannot provide your current location or a direction service.


Galway's Landmarks

Galway Cathederal

The Cathedral of Our Lady Assumed into Heaven and St Nicholas, commonly known as Galway Cathedral, is a Roman Catholic cathedral in Galway, Ireland, and is one of the largest and most impressive buildings in the city

Galway Museum

The Galway City Museum is a museum in Galway City, County Galway, Ireland. It was founded on 29 July 2006, and is located beside the Spanish Arch. The official website for the museum was launched on 27 November 2008.

Spanish Arch

The Spanish Arch in Galway city, Ireland, was originally an extension of the city wall from Martin's Tower to the bank of the Corrib, as a measure to protect the city's quays, which were located in the area once known as the Fish Market

Fig. 9. Landmarks Page(Left) and Offline Map Page(Right)

5 Software Evaluation

5.1 Testing

When developing a Phonegap or Cordova app there are many ways to test the application, each method has both advantages and disadvantages. Because there are so many ways of testing these applications, the developers are able to ensure that each competent is of sufficient quality for almost all aspect of the application before it release, minimizing the number of potential bugs in the first release of any Phonegap or Cordova application.

5.1.1 Browser Testing

Phonegap works on the browser by setting up a local application server on the user's desktop or pc, this server runs the users application as a website. When testing the application through a browser it allows us to extensively test the responsiveness of the application by resizing the browser and using automated testing tools such as selenium and firebug.

Using these tools we ran regression tests to verify web elements such as the map API was accurate every time, as well as running defect test to verify all elements were valid and in working order.

Since a browser is not the native form of the application it will not have the same feel to the application and the sizing will not match a phones dimensions exactly but it will prove useful when emulators and devices are not viable options for testing.

In terms of accuracy, when using certain features such as geolocation service the browser simply cannot match the accuracy of the phone, the browser is basing the users location off an IP server address which can be hundreds of kilometres away from the user and if the user has a VPN enable then geolocation services are virtually useless, the phone on the other hand uses built in sensors which are accurate to 10 meters.

When using browsers performance verified on different browsers, certain features would work on Chrome but not Mozilla Firefox and vice versa. Another problem with using the browser over an actual device is the speed, when running the device from a browser it tends to be much slower than using a device as the browser must run everything through the server and unpackaged it before loading the application onto the browser.

5.1.2 Emulator

Emulators work by using a virtual mobile device from an SDK. By using various different virtual devices we were able to test our application on many android operating systems including Honeycomb, Ice-cream sandwich, Kit-Kat, and Lollipop. We were also able to test the IOS platforms from operating system 5.0.0 upwards.

The reason for testing on several different Operating Systems is to test application on various screen sizes and to see if there any major difference between each device Operating System

The main advantage of this form of testing is that we can test the performance and compatibility on each type of device and search for inconsistencies. However there are a few disadvantages. The GPS get the current co-ordinates from the PC and cant use the emulated devices' GPS sensors which does result in inaccuracies. Without the use of a powerful PC, running a virtual device tends to be quite slow compared to a physical device.

5.1.3 Local Device

Testing a Phonegap or Cordova application on a physical device works by tethering a device in developer mode to a computer with the relevant software development kit and packaging the app directly onto a users device from the Phonegap command prompt rather than running the application through a server or emulator. This method of testing tends to be better in most areas.

With respect to speed the application did run much quicker than all other forms of testing, the maps were much faster than in other forms of testing, when using audio controls the device played instantly where as on the servers and the emulator it took several minutes to load the files. The geolocation features where much more accurate on a device than on browsers and emulators. The reason for this is that most phones GPS sensors are extremely accurate mainly due to the fact that it can be a useful feature for app development but tends not to be as accurate in laptops or pc because there is not much uses for these features.

By using an actual device we were able to extensive test all elements of the application and verify that everything was working as expected and that all features where as accurate as possible. Every button and control was verified and confirmed to be in working order. The Geoloaction features were tested from several areas to ensure the correct location was displayed. Offline test where performed to ensure the results are as expected.

This method of testing is very useful for Android and Windows, but when you want to test an apple device you must have an apple developers license or seek a different method. Phonegap does however have an IOS app which allows developers to run and test their app through a local Phonegap server. The only problem with this method other than IOS testing is when testing on different formats the only devices you can test are the one which are available to you, however this problem can be overcome with the use of emulators.

5.1.4 Phonegap Application Server

Phonegap's application server is a free app available on each of the main platforms marketplace where you can test your Phonegap or Cordova apps. It works by running a local server from the users computer, the user then enters the Phonegap host IP address within the Phonegap mobile app, Phonegap then repackages the app for the clients device and runs the app from there. Setup, only way to test apple w/o license, this vs local device, pros cons

5.2 Benchmarks

5.2.1 Responsiveness

Since the design of the application is based on responsive technology we expect the application to fit any screen. We would like the app to look equally as good on a tablet in landscape mode as it would on a narrow phone in portrait mode. The application has been tested on several platforms from phones to tablet, with the app responding to the shapes and sizes of each of the devices tested

5.2.2 GPS Accuracy

We expect the GPS to be accurate to within 15 meters on all devices. The location features have been tested on multiple phones such as Windows and Android devices and both platforms tend to offer accurate GPS positioning, and we believe this to be a very realistic benchmark.

5.2.3 Performance Speeds

We expect the overall performance to be very quick. All elements which do not require internet should be load instantly, while the internet elements such as the maps within the application should load within 5 seconds. We believe these to be realistic benchmarks as there is not much processing power required. None of the internet elements exceed one megabyte so even with poor wifi or mobile data, the application should still load quickly

5.2.4 Crashing

The application should only crash when the fault is due to Hardware and not the actual app. This is a realistic benchmark considering the application is only running HTML5 and JavaScript. The code has been tested several time and no errors are present as of yet.

6 Conclusions

Conclusion We set out to design a responsive multi platform mobile application, capable of working on all major platforms using Ionics framework. We accomplished this through researching other frameworks such as bootstrap and jquerey mobile. Through this research we found templates and methods of achieving our goals in developing the application. While reading research papers we were able to rule out certain frameworks and find the best possible method of achieving a truely a responsive design.

Even though we both had experience for developing a native programs such as Java for Android development and C# for Windows device development, we both have little experience in JavaScrips and Angular JS we did find it quite simple to formalities ourselves with cross-platform framework. We researched the Phonegap Beginners Guide which is available through the Phonegap official website, using this guide provided us with proficient skills to develop a basic model of the project before going through more extensive development of the application.

Over time our JavaScript skills grew immensely as we became more familiar with the language and technology. Phonegap development does have many advantages, it allows us to create multi platform applications much quicker than any other type of development, apps can be tested on any device using a Phonegap server. There are however a few disadvantages of Phonegap development including the size limit of the apps, without a license the maximum size for any Phonegap app is 50 mb which does cause problems if the app is using high quality graphics. Certain API's cannot be accessed using Phonegap, this is mainly due to the fact that different phones have different features. Developing a native app tends to be faster as the code is optimized for that device rather than all platforms for all devices.

Through the development process of this responsive application we the developers both greatly improved our JavaScript development skills and have essentailly learned a new language from this project. Since JavaScript is such a widely used language and responsive multi-platform development will have a bright future in software, we believe this project to have been a great use of time and will greatly benefit us in the future.

6.1 Recommendations

Through development of this application we realized that we made the correct decision in choosing to use the Ionic framework over other responsive frameworks such as JQuerey Mobile and Bootstrap as this framework provides more relevant libraries and templates to aid in the design of the application.

We would strongly recommend that when developing a multi platform application that Phonegap be used and that the developers plan ahead with regards to the overall size and scope of the project due to the size limit with a non-license version of Phonegap. We would also recommend that inexperienced users should research JavaScript development for beginning the development and design phased. For inexperienced developer interested in app development i would recommend they learn JavaScript and Phonegap as it give developers a wide array of options for developing future apps.

7 Appendicies

7.1 Installation Instructions

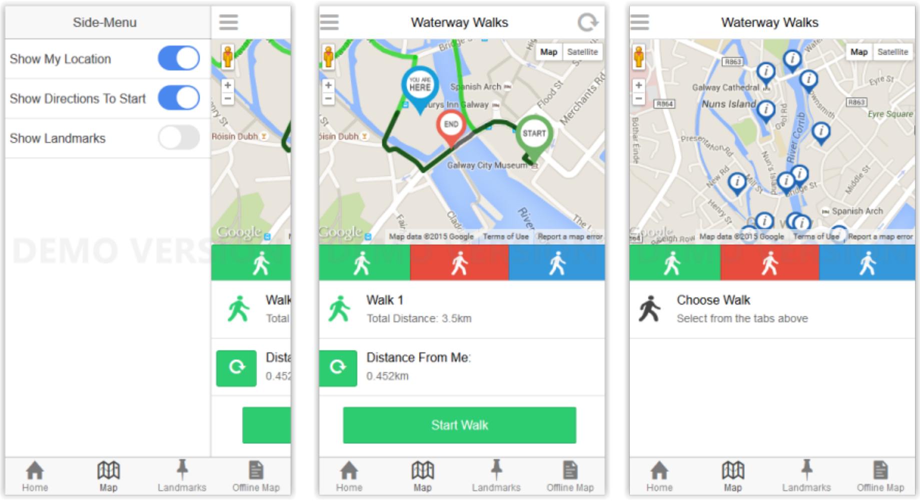
7.1.1 Installation Instructions for Android phone

1. Install drivers for the phone. These are usually found on the phone manufactures website.
2. Connect the phone to the computer via USB cable.
3. Mount the phone by pulling down the status bar which is at the top of the devicess screen and select USB Connected, then select Mount.
4. Navigate to the phones SD card. This can be done by using either the autoplay window which will pop up when the phone is connected to the computer or may be done by using the file explore.
5. Copy the APK file(CordovaApp-debug.apk) from the computer to the phones SD card.
6. Unmount by right-clicking on the phone icon in the system tray and Uplug the phone.
7. On the phone, go to setting, from there go to applications or apps depending on the phone, allow unknown sources to run by checking the unknown sources box.
8. Use a file manager application on your phone to locate the app APK. Once the file is found open the APK file
9. When the file is opened click install. The app will then run

7.1.2 Installation Instructions for Windows phone

1. Follow steps 1 - 3 from android instructions
2. On the phone, go to Store.
3. Tap the More button, then Install local apps.
4. Select the app you wish to install and tap Install.

7.2 Screenshots



Side-Menu

Show My Location: on

Show Directions To Start: on

Show Landmarks: off

Waterway Walks

Walk 1
Total Distance: 3.5km

Distance From Me:
0.452km

Choose Walk
Select from the tabs above

Fig. 10. Side-Menu(Left), Direction Service(Middle) and Show Landmarks(Right)

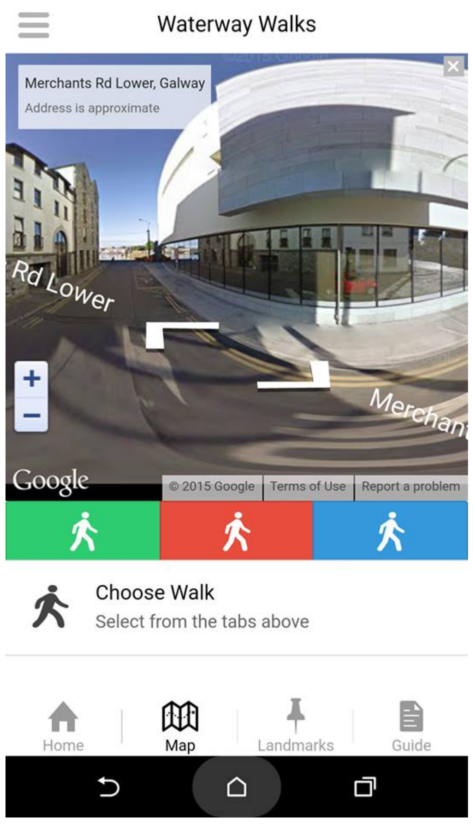


Fig. 11. Screenshot

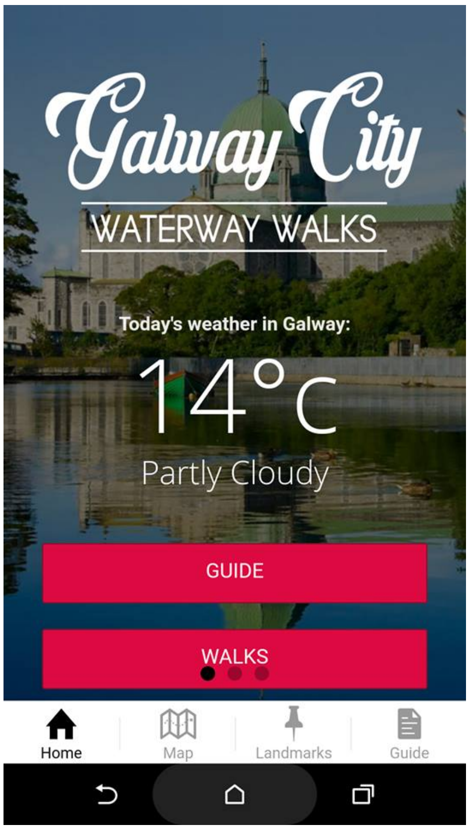


Fig. 12. Screenshot

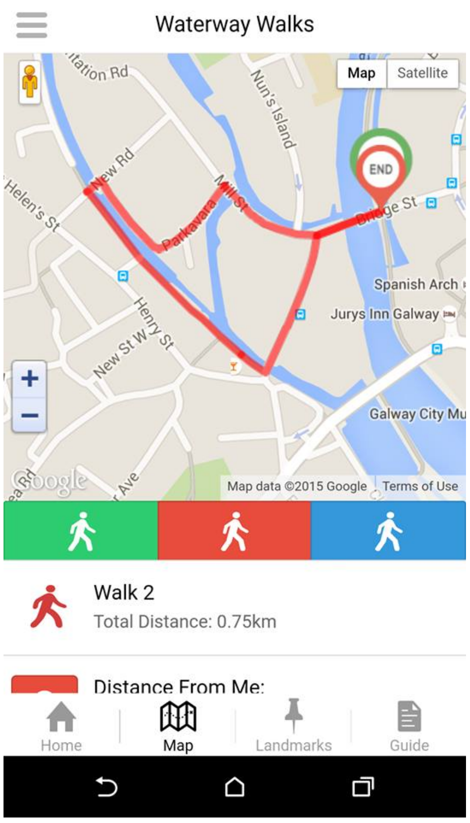


Fig. 13. Screenshot

Bibliography

[1] Heiko Desruelle, Dieter Blomme, and Frank Gielen. Adaptive Mobile Web Applications: A Quantitative Evaluation Approach. *Lecture Notes in Computer Science Volume 6757, 2011, pp 375-378)*, 2004.

[2] Heiko Desruelle, Dieter Blomme, and Frank Gielen. Performance Implications of Responsive Design. *Performance Implications of Responsive Design Book Contribution*, 2012.

[3] Joe Dysart. One size, few fits? Responsive websites may respond way too much. *ABA Journal. 100.5 (May 2014): p28*, 2014.

[4] Joe Dysart. One size, few fits? Responsive websites may respond way too much. *ABA Journal. 100.5 (May 2014): p28*, 2014.

[5] Jon Fingas. Two-thirds of Americans now have smartphones. 2014. URL `http://www.engadget.com/2014/02/11/two-thirds-of-americans-now-have-smartphones/`.

[6] Murielle Florins and Jean Vanderdonck. Mobile Web Apps. *Software, IEEE (Volume:30 , Issue: 5)*, 2004.

[7] Robert Fox. Being Responsive. *OCLC Systems & Services: International digital library perspectives Volume 28, Issue 3*, 2012.

[8] Karine Joly. One design to rule them all? Responsive web design in higher education. *University Business. 15.2 (Feb. 2012)*, 2012.

[9] Bohyun Kim. The Role of Open Web Standards for Website Development Adhering to the One Web Vision. *Library Technology Reports (August-September 2013)*, 2012.

[10] Devita Mira Lestari, Dadan Hardianto, and Nizar Hidayanto. Analysis of User Experience Quality on Responsive Web Design from its Informative Perspective. *International Journal of Software Engineering and Its Application*, 2014.

[11] Ethan Marcotte. Responsive Web Design. 2010. URL `http://alistapart.com/article/responsive-web-design`.

[12] Nebeling Michael and Moira C Norrie. Responsive Design and Development: Methods, Technologies and Current Issues. *Web Engineering Lecture Notes in Computer Science Volume 7977, 2013, pp 510-513*, 2013.

[13] S. Mohorovicic. Implementing responsive web design for enhanced web presence. In *2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2013.

[14] JaeWon Moon. Advanced Responsive Web Framework based on MPEG-21. *2012 IEEE Second International Conference on Consumer Electronics*, 2012.

[15] Kailashkumar V. Natda. Responsive Web Design. *An International Referred Journal of Business, Accounting, Information Technology & Law Vol 1, No 1 (2013)*, 2013.

[16] James OToole. Mobile apps overtake PC Internet usage in U.S. 2014. URL `http://money.cnn.com/2014/02/28/technology/mobile/mobile-apps-internet`.

[17] Serena Pastore. The Role of Open Web Standards for Website Development Adhering to the One Web Vision. *International Journal of Engineering and Technology Volume 2 No. 11, November, 2012*, 2012.

[18] Nicolas Serrano, Josune Hernantes, and Gorka Gallardo. Mobile Web Apps. *Software, IEEE (Volume:30 , Issue: 5)*, 2013.

[19] Sujan Shrestha. Mobile web browsing: usability study. *Mobility '07 Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology, ACM*, 2007.

[20] Jeremy Snell. Flexible everything: getting responsive with web design. *Computers in Libraries. 33.3 (Apr. 2013)*, 2013.

[21] Zachary Thacher. Responsive Mobile Site Design. *Global Cosmetic Industry. Apr2014, Vol. 182 Issue 3*, 2014.

[22] Jeff Wisniewski. Responsive design. *Online Searcher. 37.1 (January-February 2013)*, 2013.

[23] W3C World Wide Web Consortium. World Wide Web Consortium, W3C. 2014. URL `www.W3C.org`.

[24] Bin Zhu. Responsive Design: e-Learning Site Transformation. In *Networking and Distributed Computing (ICNDC), 2013 Fourth International Conference on*, 2013.