Miscellaneous Reports

# Test Report

The testing for this project was conducted on the web browser and on mobile phones. The Apache Cordova framework facilitates testing and deployment of applications because of the fact that it is written in HTML, JavaScript and CSS. For my testing phase, I used Google Chrome as my default web browser and a few android phones and tablets as my deployment device.

Based on the fact that the application is written in HTML, JavaScript and CSS, you can easily test a function without having to deploy the application to a virtual or physical device. For plain JavaScript functions, such as getting location, displaying a map and testing layouts, I simply used my web browser to test it. For the native mobile phone functions, such as accessing camera, displaying messages, accessing calendar, I had to deploy the app to a physical, or virtual, device in order to access the test results.

This approach was appropriate for this project. The reason why it was appropriate is due to the fact that I was using one of the best IDEs for developing Apache Cordova applications. The IDE is called JetBrains WebStorm. On this IDE, you can have direct access to virtual devices and also deployment options, making it easy to build, compile and run your code across several devices. The best useful feature about JetBrains WebStorm is the ability to create a Local Host in order to test your applications. This is how I was able to conduct testing on my Server – Client connections without always having to deploy it on a physical device.

For my PHP/MySQL scripts, I was using an IDE called JetBrains PhpStorm. This IDE is also great for writing PHP scripts, it provides code completion, refactoring and more. I chose this IDE because of its Version Control Systems integration, which enables the IDE to support remote deployment, databases/SQL, REST Clients and other tools. It also takes care of the Code Quality Analysis to help the developer write neat code that’s easy to support.

This approach was simpler, because I only had to test connections and queries to my remote Database, which is stored in a web server called 24Hosting. Once I knew the script was working, I deployed it to my web server and then accessed it from the Church Finder Ireland application.

The most interesting test happened when I sent the application to a friend over in Brazil. I needed to test the speed of the application when accessed from another country. Since my web server is located in England, the connections took longer than expected when compared to connections made from Ireland. This phase of the testing only used the *Date().getTime()* method to get the milliseconds that a function takes to execute.

The table below shows the difference between accessing my web server from two different locations:

|  |  |  |
| --- | --- | --- |
| **Name of function** | **Brazil (in seconds…)** | **Ireland (in seconds…)** |
| Accessing list of Churches | 3.1 | 1.2 |
| Getting details of a Church | 1.9 | 0.8 |
| Accessing map | 1.7 | 1.8 |
| Uploading a picture | 2.1 | 0.6 |
| Viewing an existing Picture | 2.6 | 1.3 |

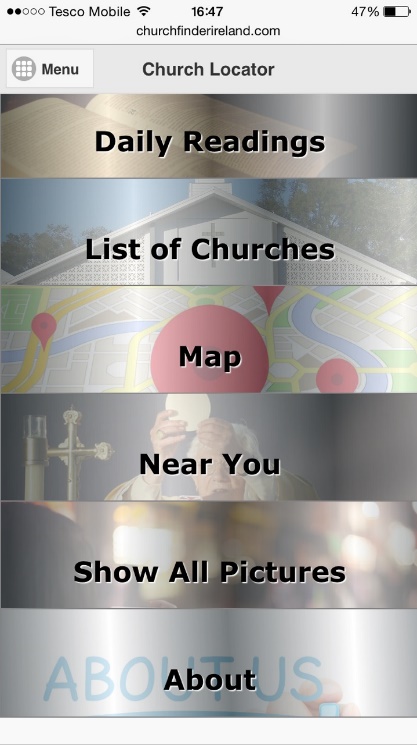
When the application was finished with the development, I initialized the system testing. This was an enjoyable approach because I was able to identify the main problem when developing Apache Cordova applications.

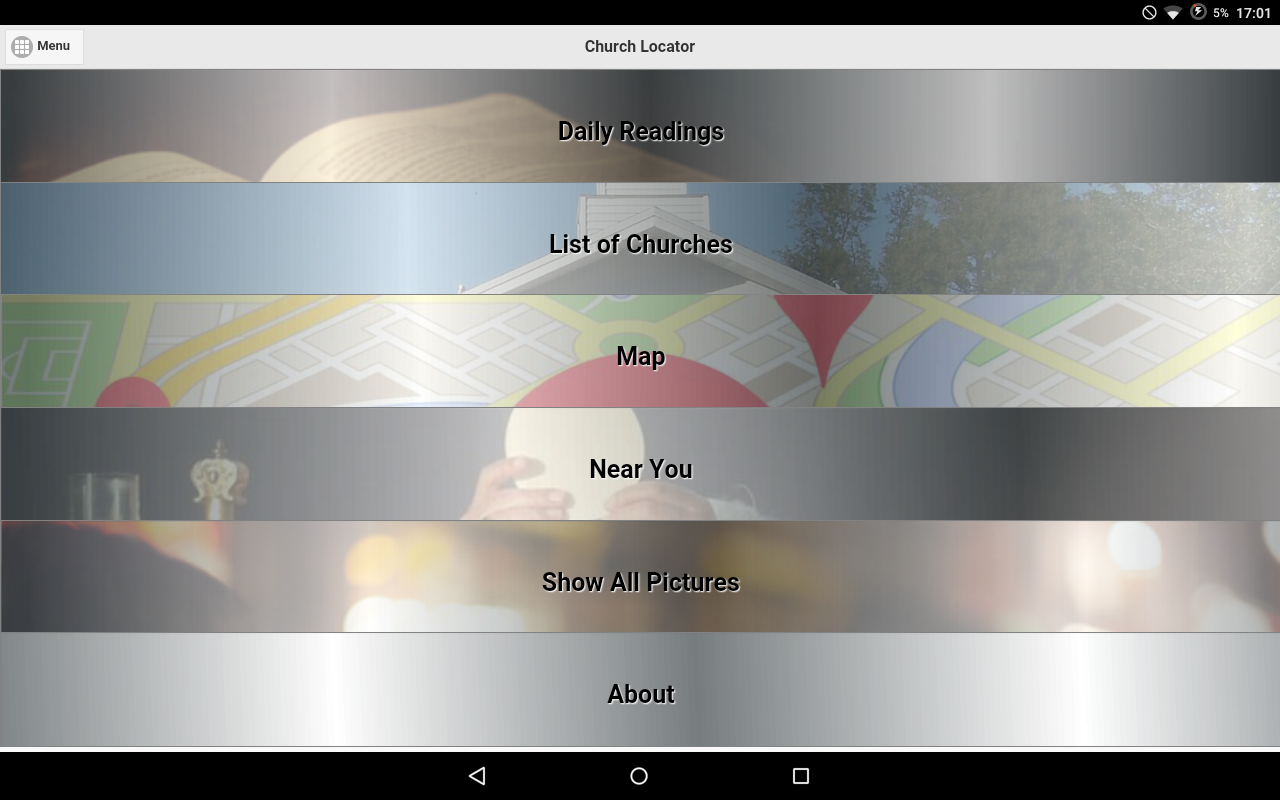
The table below shows a list of devices, and an indication of whether the application was successfully tested on the device:

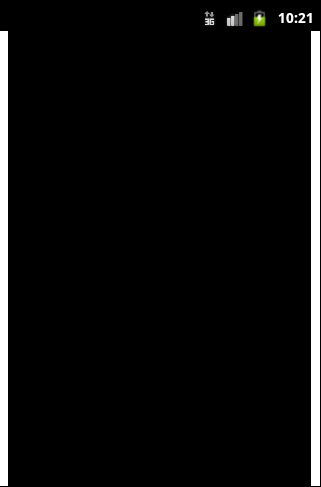
|  |  |  |  |
| --- | --- | --- | --- |
| **Device Name** | **OS Version** | **Successful** | **Comments** |
| Samsung Galaxy S5 | Android 5.1 | Yes |  |
| Google Nexus 5 | Android 5.1 | Yes |  |
| Motorola Moto G | Android 5.1 | Yes |  |
| HTC One | Android 5 | Yes with errors | CSS content not being correctly rendered, making the layout of the application really messy |
| Sony Xperia SP C5303 | Android 4.2 | No | The application was unable to start. Unknown problem |
| Samsung Galaxy S4 | Android 4.4 | Yes with errors | CSS content not being correctly rendered, making the layout of the application really messy |
| Samsung Galaxy S3 | Android 4.4 | No | The application was unable to start. Unknown problem |
| Samsung Galaxy Tab 2 10.1 | Android 4.4 | Yes with errors | CSS content not being correctly rendered, making the layout of the application really messy |
| Samsung Galaxy Pro 4 10.1 | Android 5.1 | Yes |  |
| iPhone 6 | iOS 8 | Yes |  |
| iPhone 5S | iOS 8 | Yes |  |
| Desktop | Windows 8.1 | Yes with errors | Mobile native functions do not work |

The images below are an illustration of the system testing made on physical devices:

**Samsung Galaxy S5 iPhone 6**

****

**Sony Xperia SP C5303 Galaxy Tab Pro 4 10.1**

****

I was able to identify two major problems with my application by conducting the system testing.

The first problem is with the Android version. Whenever I tried to run my application on devices that had Android 4.4 or lower, the application either displayed incorrectly or did not even start. The reason is still unknown, as I haven’t been able to identify the problem yet.

The second problem is with the screen size. This application was tested mostly on 5’ inches devices. The CSS rendered properly on 5’ inches devices and also on devices running Android 5 or 5.1, but it did not run properly on lower API’s or on smaller screen sizes. From research, I found out that the developer has to specify CSS-media-queries to resize the application according to the actual resolution of the device. This will be implemented in future versions of the application.

The most difficult part of the testing phase occurred when I was testing the GPS functions in order to get the routes between two points (the user location and Church location). In order to achieve this, I had to deploy the application to the phone every time I changed a line of code, and this was difficult to handle. After several tries, I was able to get the function working properly on the testing device.

Unfortunately, I did not have time to use testing frameworks, such as QUnit (for JavaScript unit testing) and PHPUnit (for PHP unit testing).

# Installation Manual

This installation manual will cover everything that is needed for a developer to compile, run and deploy this application on a mobile device.  
 **Operating System:** Microsoft Windows 8.1 / Windows 8 / Windows 7  
**Programs needed:** Github / JetBrains WebStorm / Notepad++  
**Plugins and SDKs:** JDK / Android SDK / Apache Cordova / Node.js

The first step is to clone my application source code folder on Github to your desktop. Open your web browser, navigate to *https://windows.github.com* and download Github for Windows. Once the download is finished and you have installed the program, you will get 2 new icons on your desktop: Github and Git Shell. For this tutorial, we will be using the Git Shell.  
Once the Shell is opened and loaded, type “**git clone**”and then enter the project’s repository page, which is [**https://github.com/x00093357/4th\_Year\_Project.git**](https://github.com/x00093357/4th_Year_Project.git). The complete command should look like this:   
**>** git clone <https://github.com/x00093357/4th_Year_Project.git>  
Once this task is finished, open the Windows Explorer and go to your Documents – GitHub. The clone of the repository should be there.

Now that we have a copy of the repository, the next step is to get the IDEs and frameworks needed to compile, run and deploy the application. We will begin by installing Apache Cordova on our computer.  
To install Apache Cordova on your computer, follow these steps:

1. Download and install Node.js (<https://nodejs.org>). Following installation, you should be able to invoke **node** and **npm** on your command line.
2. Download and install a Git Client (<http://git-scm.com>). Following installation, you should be able to invoke **git** on your command line. We won’t be using GIT directly, but Apache Cordova uses it behind-the-scenes to download plugins.
3. Install the **cordova** module using **npm** utility of Node.js. On your command line, enter: **npm install –g cordova**.

After successfully following these instructions, you should have the Apache Cordova framework installed on your computer. The next step is to download the Android SDK and the JDK (Java Development Kit).

You can download the Android SDK by navigating to the Developer Android website (<https://developer.android.com/sdk/index.html>). You do not need Android Studio for this tutorial, but the installation should take a long time to be completed.

You can download the JDK by navigating to the Oracle website (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>). Make sure to select the right version of your Operating System to prevent any errors.

Once you have download and install the Android SDK and the JDK, you will be able to compile, run and deploy Apache Cordova applications.

This project does not require MySQL or PHP to be installed on our local machines, since these files are located on the remote web server. But if you want to access the files locally instead of remotely, you can download MySQL by going to this website:   
- http://www.mysql.com/downloads/  
and you can download PHP by going to this website:   
- <http://php.net/downloads.php>

Now that we have all the plugins, frameworks and APIs, the last thing we need is an IDE. For this tutorial, I will be using JetBrains WebStorm. The program has a 30-day free trial and if you are a student, you can have a student license for 365 days.

You can download JetBrains WebStorm by going to this website:   
- <https://www.jetbrains.com/webstorm/download/>

Once you have installed JetBrains WebStorm, you can open this project by going to File – Open Project and selecting the cloned GitHub folder (Documents – GitHub).

The HTML, JavaScript and CSS files will be located on the “www” folder (ChurchFinderIreland – www).

Next step is to select a physical or virtual device. Go to the “Run” menu, and select “Edit configurations”. If you want to run the application on a virtual device, change the “Command” option from “run” to “emulate”. Also make sure you have installed an Android Virtual Device (you can accomplish this while installing the Android SDK).

If you want to deploy the project on a physical device, the “Command” option should be “run”. Also make sure you have installed the appropriate drivers for your device on your computer and that you have enabled “Developer Options” and “USB Debugging Mode” on your Android device.

To compile and run this application, select “Run” on the navigation menu, and then select “Run” or press Shift + F10. If you correctly installed all the plugins and SDKs, the project should compile and get deployed on your selected device.

# User Manual

# Post-Project Review