Canals and River Trust
Numericanals App for iOS and Android

Code and Development Report

mxData Development

Produced by	Nick Hill & Simon Sturge
Date	23/09/2015
Version	V 0.1
Document status	
File location	

1 Document Version Control

Number	Date Issued	Comments	Author
V0.1	23/09/2015		Nick Hill / Simon Sturge

Contents

2 Glossary

A glossary of terms will be included where appropriate

ObjC	ObjectiveC

3 Overview

This high level report outlines the actions taken to localise the Numbericanals App from French to English. It also describes some of the issues found, improvements that we feel may help others do this task and finally describes what our observations of the code base were in general.

4 Observations and Development

4.1 Development

4.1.1 Localisation

2. Localising the Numericanals app was a fairly simplistic task. Both iOS and Android provide mechanisms for using different languages in an application. To localise an iOS app, every user visible word or sentence needs to be wrapped in a simple macro, and then included in a single localised strings file. A similar approach is taken in Android, where the strings are defined in a localised file, and then given an ID to use when developing the app. In both cases, the developer who initially created the Numericanals app followed these approaches. This meant that including English translations for both applications was as simple as the following steps:
   a. Create a new localised strings file for the specified language
   b. Copy the strings from the original strings file
   c. Translate the string values into the specified language

   No problems were encountered when following the above steps, and both apps were localised without any complications.

4.2 Observations

4.2.1 The following list includes both good and bad points that were found in the iOS app:

a. Minimum OS is 7.1 - ability can reach 96% of users
b. Universal – ability to run on iPhones and iPad's
c. No Swift use – Fully written in ObjC, ideally would be written in the more modern Swift language
d. Storyboards – Interface visually created, no code interface creation; this is good
e. Constraints – Size and position of user interface elements defined by constraints
f. Core Data – Persistent storage was Core Data, this is overkill and a simpler approach could be suggested

The only problem encountered with the iOS app was location management, in which it hasn't been updated to use iOS 8's location permissions.

4.2.2 The following list includes both good and bad points that were found in the Android Client:

g. Minimum OS is 4.0 – ability to reach 95% of users
h. Third part code – Uses some third party code to manage server responses which was outdated
i. Custom Controls – Use of many custom user interface controls. This is a concern, and we would suggest removing them and opting for a more standard interface
j. Location Management – Correct use of location services as defined in the Android docs
k. Support Library – The support library in use was out of date

5 Summary

5.1 Summary and Developer Notes

5.1.1 Generally speaking, the code base for both applications was very respectable. In some cases, such as persistent storage and JSON parsing, the code looks to have been over complicated to a point at which it’s very hard to manipulate and change how it works. Simplifying the code e.g. moving from CoreData to NSUserDefaults, and moving to a more standard approach of JSON parsing could improve these areas.

5.1.2 In terms of improving the code, it would be beneficial for both the end user and the developer to remove all code that relates to the user interface. In its current state the apps include third party code to display non-standard interface components, and complex code to manage the transition animations. Removing these libraries, and moving to standard components and transitions would make development easier and result in a more pleasant interface for the user.