Report for Currency Trader news alert app

* 350 words per hour
* 6000 divided by 350 = 17 hours
* 4 hours per day = 4 days = 16 hours Wed / Thurs / Fri / Sat

Potential Topics:

* Planning – sprints proof of concept apps – separately & within the project
* naming conventions.
* ratio of time spent reading vs. writing is well over 10:1. We are constantly reading old code as part of the effort to write new code. (uncle bob)
* ~~Testing~~
* ~~Threading – future development.~~
* Localisation
* ~~Time zones & time differences~~
* AppCompat for themes / toolbars /
* Testing on multiple phones – online test suite ????
* Xamarin . other unit tests ????????
* Custom Adapter vs Recycler Adapter & Card view (notify adapter issue)
* Graphics – custom splash screen – flags – different sizes using website
* using Android Asset Studio: <https://romannurik.github.io/AndroidAssetStudio/>
  + <https://www.colorhexa.com/0070bf>

Passing data between activities – using intents – vs json – vs method arguments and parameters – vs setting a property on the receiving class

Shared preferences for date of last data update

* Database – how things are stored – ORM – Ignore Item (Ticks issues) - Ticks / DateTime object / Sqlite
* Export watch-list to csv file – also have an import csv option.
* Integrate with Google calendar (API ?) – flag user time scheduling conflicts.
* Schema for ‘forex-factory’ xml download – program to this (?).

App crashed when deployed to real phone – later discovered it crashed on phone & simulator when running app for the 1st time (deleted app each time to be sure). Method was trying to access database before table was set up. 16/July

Date of xml update / download is now stored using Shared Preferences 17 / july

Testing:

The importance of testing in the development of any system or application cannot be overstated. With a variety of testing methodologies to choose from I decided to implement a manual testing process for several reasons.

In our advanced programming module, we had exposure to TTD, test driven development and the work of ‘Uncle Bob’, Robert Martin. In time, and with a more experienced skill set, TTD is something I would aspire to, writing your tests before your code seems to be a very progressive way to approach development.

We also had exposure to Unit Testing, a methodology which should be employed by all developers, regardless of whatever other testing methods are being utilised at higher levels within their team or organisation. This requires that your code be developed in such a way as to facilitate such testing.

In my application I have endeavoured to write my code to enable unit testing, where possible writing my methods and functions so that they return a value such as a true or false bool, or an int containing the number of rows updated in the database etc. I have also ‘wired up’ a unit test project within Visual Studio, with some dummy tests.

A large percentage of my application uses technologies that I’ve researched myself, e.g. utilising Android’s date and time pickers, phone notifications etc. As result, it has been impractical to write tests, or adapt my code ahead of time for such features. I would though, regard this as a priority for any future development.

Manual Testing:

During development, anytime I would introduce a new feature, e.g. adding an extra item to a menu, I would go through a process of not only trying out the new feature, but also verifying that all the previously working features still worked as expected. Choosing manual testing, in the end, was a process of documenting my natural, personal development style.

I researched a lot of tutorials on YouTube, where different tutors presented how they laid out their manual tests in an Excel format, and took that as a basis for my own Excel based, manual test layout.

Testing on multiple devices:

For future development of an application that is aiming to be released commercially it would be desirable to test the app on as many different devices as possible, to access both functional performance and visual aesthetics. While it is possible to run many different emulators in Visual Studio, a time-consuming task, a commercial option such as Microsoft’s own ‘Visual Studio App Centre’ which includes Xamarin Test Cloud, (appcenter.ms) would be helpful. This lets the developer test their app ‘in a hosted device lab with 1000s of real iOS and Android devices. You’ll receive test results, full-resolution screenshots of every step, along with performance metrics’.

Please note, manual test cases are included in an accompanying excel file.

Xamarin.UITest is a testing framework that enables Automated UI Acceptance Tests written in [NUnit](http://www.nunit.org/) to be run against iOS and Android applications. It integrates tightly with Xamarin.iOS and Xamarin.Android projects but it can also be used with iOS and Android projects written natively in Objective-C/Swift and Java. Xamarin.UITest is the Automation Library that allows the NUnit tests to execute on Android and iOS devices. The tests interact with the user interface just as a user would, enter text, tapping buttons, and performing gestures – such as swipes. For more information on the components enable Xamarin.UITest to automate a mobile application, please read about the [core concepts](https://docs.microsoft.com/en-us/appcenter/test-cloud/core-concepts).

Threading:

I implemented threading in the following methods in UserAlertsActivity;

* SetAlarm
* DeleteAlarm

These methods both use the AlarmManager class to set and delete alarms. It was possible for me to implement these on separate threads as the application didn’t need to wait for any returned response before continuing. If they remained on the UI thread, the UI would freeze until they had completed their work. Also, the debug console window was warning that there were too many processes running on the UI thread.

I also tried to implement threading in the ‘Update XML’ option (Main screen, top menu, 1st option). When the user selects this option, the application downloads an XML file from the ForexFactory.com brokerage site. During the download, the UI temporarily freezes, preventing any other user interaction with the application. I experimented with putting this process onto a separate thread, which did prevent the UI from freezing during the download, but it caused other issues downstream because I hadn’t designed the application with threading in mind from the outset, e.g. methods were getting called before the required data that was returned by the download thread was available, causing timing issues in terms of updating the screen display and database access

In any future development I would like to implement threading throughout the application. While threading would bring performance benefits it would require some considerable change to the architecture of the application to avoid race conditions etc.

Geographic and location issues:

Currency trading is a global activity and the website that provides the XML file of weekly market news events, ForexFactory.com is an American site. To simplify zone type issues, they release their XML in GMT, Greenwich Mean Time.

During DST, daylight savings time, we are one hour ahead of GMT time.

In method:

ConvertXmlAndStoreInDatabase of DataStore.cs

I check to see if the application is currently in DST using:

DateTime.IsDaylightSavingTime Method ()

If it is, then an hour is added to the xml data as it is stored in the database to bring it in line with DST. When not in DST nothing is added, so the time simply remains the same. This only applies to market events and alerts and does not apply to personal alerts set by the user.

Currently the application would only be suitable for countries that are in the same time zone as Ireland and the UK, and that follow DST. In future development a facility could be added to update the market alert times to be correct for whatever time zone the device running the application is in.

Device language setting:

Another issue that presented itself very unexpectedly was that of which version of English the host device is set to. During development I have been testing the app on both my own phone, Samsung S7 (Oreo) and the emulator (Nougat). Unknowingly the Samsung had its language set to English (Ireland) while the emulator was set to English (American).

A method that used I used in ConvertString\_s\_ToDateTimeObject:

DateTime.Parse(dateAndTimeString)

to convert a string to a date-time object, caused the app to app to fail on the Samsung phone, a difficult problem that took some time to locate and solve, and was eventually fixed by creating a CultureInfo object:

DateTime.Parse(dateAndTimeString, new CultureInfo("en-US"));

Known issues:

If a user alert is set on market data whose date-time has already passed, then the alert fires instantly.

* A check needs to be added to only allow the user to set alerts on future market events.

Multiple alerts are allowed for the same date-time.

The ‘Update Market Data’ (Main Activity top menu, option 1) hangs if there is a problem with the host device’s wifi or internet connection. An option would be to have a timeout type function surrounding the xml download, and to use test data already in the Assets folder until downloading is possible again.

Report Ideas:

Workflow is more enjoyable when you have a definite plan

that is laid out in a logical progression - step by step!

Takes the 'creative worry' out of not knowing what you are going to work on next - helps to overcome procrastination!

Learning experience - e.g. 2 different ways of implementing Toolbar using appcompat

- now have code available for use in future projects.

Refactoring & redesigning naming & labelling conventions for entire project including;

* + - method calls & activity names
    - @string (needed for localisation)

Preferences – added an option so the user can adjust all the market times plus or minus 59 minutes – this updates all data shown in Main Activity – BUT it does require a new download of the xml data – I would prefer to just adjust the already downloaded data that is stored in the database – big job = FUTURE DEVELOPMENT!

Personal Alerts are left at whatever time the user set them for – they are NEVER altered.