Milestone 1: Set up tools

Following the integration of the Xamarin.UITests tool, I will select the targeted functionalities to base my unit tests on. I will also set up Exerciser Monkey for my application. The time, effort, and complexity for both tasks will be tracked

Tool #1: Xamarin.UITests

Background Information:

Xamarin.UITest automates the user interface by activating controls on the screen and providing input anywhere a user would normally interact with the application.
To enable tests that can *press a button* or *enter text in a box* the test code will need a way to identify the controls on the screen.

To enable the UITest code to reference controls, each control needs a unique identifier. We set this identifier by using the AutomationId property to all controls that are required for testing. For example:

```
<Button x:Name="b" AutomationId="MyButton" Text="Click me"/>
<Label x:Name="l" AutomationId="MyLabel" Text="Hello, Xamarin.Forms!" />
```

Setting Up

1. Set the ANDROID_HOME environment variable (Windows) with the path to the Android SDK
2. Ensure that the app requests the INTERNET permission by inserting this snippet in the AndroidManifest.xml

   <uses-permission android:name="android.permission.INTERNET" >
3. Disable **Use Shared Mono Runtime** (this will prevent Xamarin.UITests from running in App Center for Test, and the CLI will throw the error "Mono Shared Runtime is not supported") by going to project properties -> Android Options -> Packaging Properties -> uncheck Use Shared Runtime.
   a. This feature is usually enabled to make debugging faster with each round, and is left on when not using the UITest tool
4. Set an automation id to all elements to control (see above)
5. Right-click the solution, select File > New Project
   a. Search for Xamarin.UITest Cross-Platform Test Project
   b. Be sure to select add to existing solution
6. The new project has two classes in it: AppInitializer and Tests.
   a. AppInitializer contains code to help in test running, platform, and what the app exactly needs to be used. Change the line in AppInitializer to `return ConfigureApp.Android.InstalledApp($packagename).StartApp();`
   b. Tests contains all the tests you write, and will screenshot the results. Examples:

   ```
   app.Query(c=>c.Marked("MyButton"))
   ```

```
app.WaitForElement(c => c.Marked("Add"));
app.Tap(c => c.Marked("Add"));
app.WaitForElement(c => c.Marked("Cancel"));
app.Tap(c => c.Marked("Cancel"));
```

How to Run

After writing your tests, they will show up in a Test Explorer, which you just click play to run. Screenshots are taken and saved for viewing later.

Time, Effort, and Complexity of Set-up

The process of setting up Xamarin.UITests tool less than half an hour. The instructions provided by Microsoft were clear and no issues were encountered. However, familiarity with Xamarin was required for this set-up. As I already had familiarity, the process was very simple, but I imagine that a user with no prior experience may need another hour to fully understand what is going on. Overall, I would so far give this tool's usage low complexity.

Tool #2: Monkey Exerciser

Background Information

Android Monkey is a python-based testing script process that can be run by writing any specific python script or we can directly apply a test for the installed application using the command line.

Once the test process will be done then it will generate a log file of the events and crash if any. Generally, these testing tools are for the application developers while development in progress. A prerequisite for using this tool is to have Android SDK downloaded and installed on the system.

Monkey Exerciser will randomly check for user events such as clicks, touches, or gestures, as well as several system-level events.

Setting Up

1. Have Android SDK downloaded and installed
2. Set the ANDROID_HOME environment variable (Windows) with the path to the Android SDK

How to Run

1. Go to ../Android_Sdk/Platform-Tools directory
2. Execute the following command: `adb shell monkey -p yourpackageneme -v 1000 > app_log.txt`

    a. Additional arguments can be added to this command and are found at https://developer.android.com/studio/test/monkey
3. Results are stored in app_log.txt

Time, Effort, and Complexity of Set-up

The process of setting up Monkey Exerciser was almost inexistent. The steps for setting up were already completed since having an android project in the first place requires those steps. Tutorials online suggested that this was only available for Android Native projects (for example, on Android Studio), and I was concerned that with my Xamarin.Forms cross-platform project the tool wouldn't be able to recognize the application. However, I was pleasantly surprised to find that the tool ran with little issue. I had to troubleshoot due to having the server adb version mismatch the client adb version, but after resetting the adb server this issue was quickly resolved. Due to these issues, I would so far give this tool's usage a medium complexity.