Technical University of Moldova

CIM Faculty

FAF

Report

Integrated Development Environments

Laboratory work #3

Done by: student, gr. FAF-151

Bîzdîga Stanislav

Verified by: Patrașco Alex

Chișinău - 2017

Topic: Mobile Applications Development

Prerequisites:

- IDEs: **Visual Studio**, Xcode, Android Studio, Eclipse, NetBeans
- Languages: **C#**, JavaScript, Objective C, Java, Swift
- Technologies and Frameworks: Windows Mobile, iOS, Android, **Cross-platform**

Objectives:

- Basic knowledge of mobile application architecture
- Basic knowledge of specific platform SDK.

Tasks:

Choose one: (What's chosen is in bold)

for 5||6 pts:

Make a simple Hello World application with 2 buttons that will display 2 different views, with different user interaction elements.

for 7||8 pts:

You must implement a simple stopwatch or a clock alarm.

for 9||10 pts:

Pomodoro technique application, or some more sophisticated mobile application.

for *Extra* 3 pts:

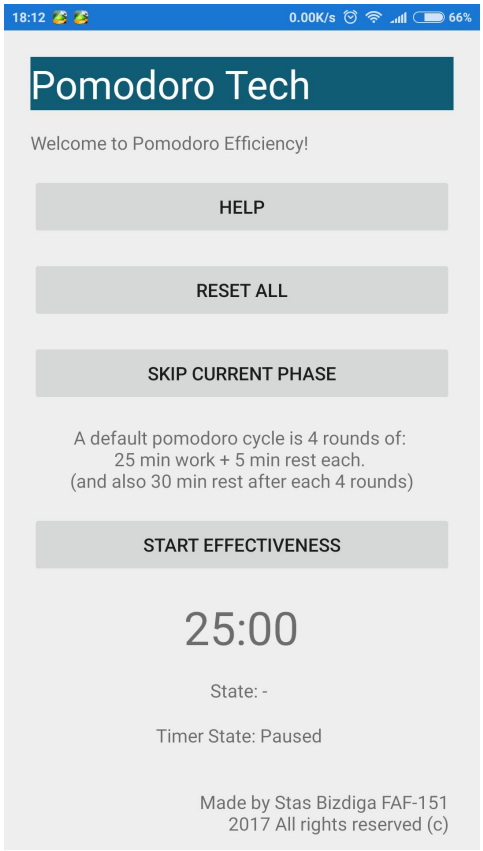
Use some cross platform package in order to make a cross platform application (you must be able to run your app on Android and iOS platform)

Work processing

The current app was an interesting challenge to do. I chose to make a pomodoro technique app that will be available for mobile platforms. For this task, It was required to have an appropriate IDE, and so I went for **Visual Studio**. I've never tried it before but heard it was good. However, setting it up took a whole lot of precious project time. Afterwards, I chose a package to work with that had cross-platform support. It's name is **Xamarin Forms**. It gave me the possibility to code in **C#** and not worry about each platform and have a common code instead, that would work perfectly with any platform. (That's what xamarin forms promised at least)

Product description:

Pictures:



^ Due tapping the RESET ALL button (or simply when starting the app)

Features:

- **Help button** - displays the explanation of what the app does and how to use it.

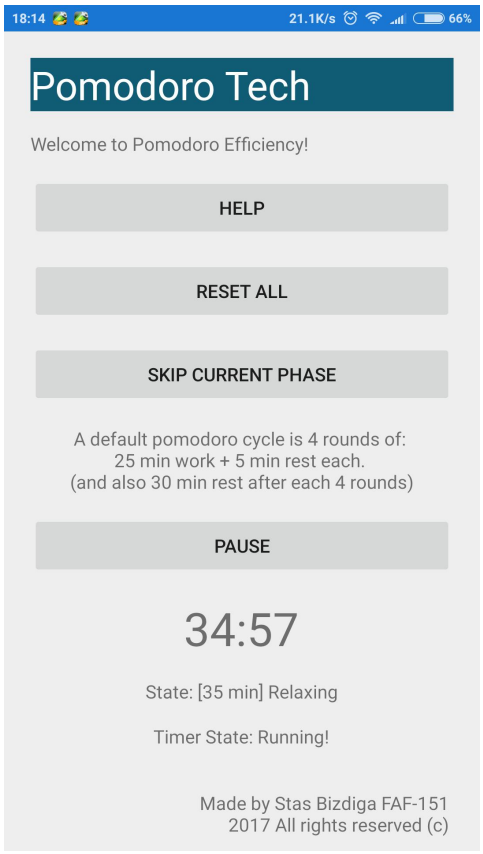
- **Reset All button** - resets the app to its starting point.

- **Skip Current Phase button** - sets the timer to zero, so that you reach the next phase/state, also helpful for testing/debugging.

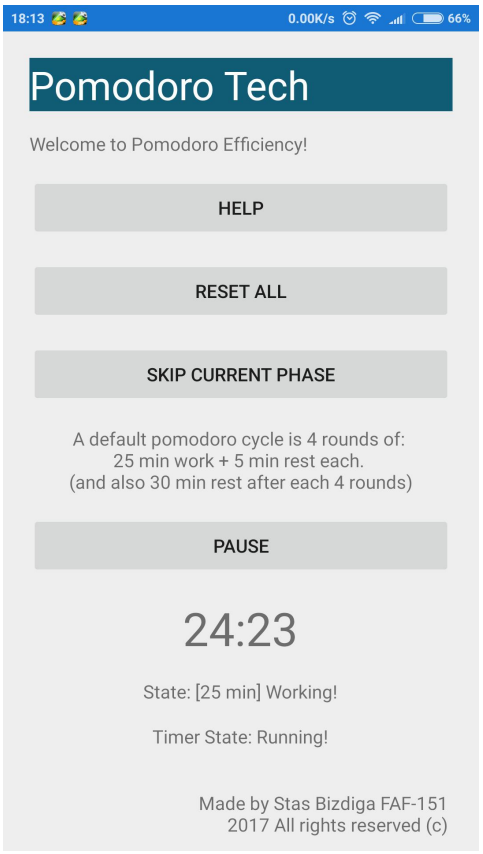
- **Start Effectiveness** - begins a pomodoro technique cycle composed of 8 phases: working 25 minutes , resting 5 minutes, working 25 minutes , resting 5 minutes, working 25 minutes , resting 5 minutes, working 25 minutes , relaxing 35 minutes.

- Due beginning the cycle you are able to pause and resume the timer if necessary. The button for starting transforms into either **resume** or **pause** .

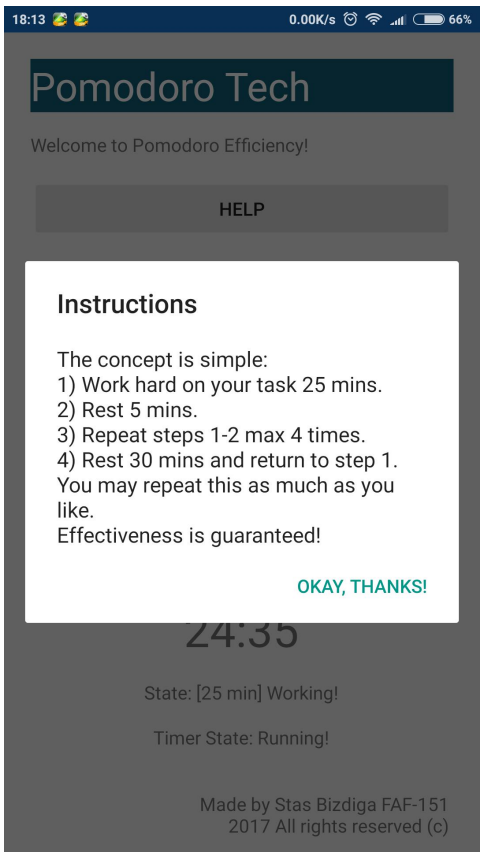
- **Time indicator** - shows how much more time you will need to work or rest until the next pomodoro phase. You will get notified when the state changes via 1 sec **vibration**.



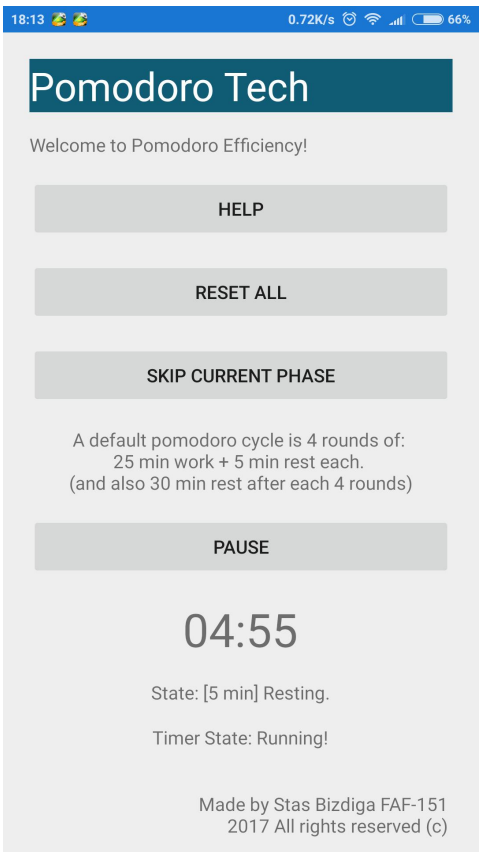
^ *Due reaching the relaxing phase*



^ *Due reaching the working phase*



^ *Due tapping on the Help Button*



^ *Due reaching the resting phase*

Notes regarding the task:

In theory it seemed good, but in practice, working with such young package was a bit of a struggle. I've learnt the syntax relatively easy but it was not very stable, and most of the project work time was spend on headaches with weird bugs regarding visual studio and xamarin forms, and not the code that I was actually writing. I learnt the IDE was buggy as well. Often times a rebuild or IDE restart would solve some "bugs" that would not be clear at all about their source.

Xamarin forms, regardless of the seemingly vast features and attractiveness turned out to be a painful disappointment. There are a lot of yet unimplemented things that are necessary in production, things like a simple timer, threading processes, sound API. All these were left as unimplemented and I had to look for external source codes and libraries - that was the only way to avoid new problems.

In order to implement the features in the app, I've taken the following approach:

I found a nice package that helped implementing the timer functionality, called "advancedTimer"

I used simple switch-case logic to handle different phases the timer has to go through and also buttons' different responses depending on the app current state.

After trying two open-source API's for handling sound and failing at both, even if I was excited since I had some nice SFX for the app ready, I realized it's just not going to work any soon so I decided to implement the vibrating function rather. It worked easily and instantly. Easy!

C# isn't as hard as i thought. Using it with .xaml was fun and intuitive. Even if I didn't program in C# before, I was able to handle it with ease.

CROSS-PLATFORM

1) Android

Now, all was clear and simple here, android was working flawlessly even though I did have the need to solve a lot of weird bugs with random approaches during the process of coding such as rebuilding the project and closing - restarting the IDE as mentioned before. Including also, really weird NET framework installing and fixing.
As I finished the project I was glad to find an APK file in the */release* folder. I dropped it into my google-drive, and installed from my smartphone. It went smooth and worked!

(Except the fact that I firstly uploaded the unsigned version of the apk which didn't want to be installed: "parsing error" it said. But thanks to Googling I found the reason and solved it shortly. There was also a signed .apk)

2) iOS

Really, no comment here. Xamarin forms assures that it should work on any mobile platform because the code is common.
(I wrote a common C# code and did not touch the other platforms' codes at all. droid works.)
I have no iOS device so I had no possibility to test the app. That's very sad, because I wasn't able to even emulate it. Turns out:
VisualStudio requires a Mac to connect to in order to emulate an iPhone. It's weird, and I tried a lot of ways to go around this problem.
I even tried to install a VirtualMachine with MacOS, which eventually failed, giving my Windows a blue screen of death each time I tried to run the MacOS. At that point I gave up, since, VisualStudio didn't ever compile the program in an .app bundle regardless my tries.
So I could not send it to a friend with iPhone.

Conclusion

This laboratory work motivated me to study about mobile programming. It provided me an opportunity to study C#. I used it only a bit before in Unity3D. Turned out it's not as hard as expected, but rather intuitive and even fun.

Also, I learned about Cross-Platform development and how to do it, i.e. how to implement the required package so that it works flawlessly among different devices. Sadly it didn't work as expected but I would be able to do it in future for sure. Xamarin Forms is great, but also terrific. I'm looking forward to see it polished up in the near future.

Visual Studio really was disappointing, a simple emulator would run really slowly and lag out my laptop, thus I had to get to my desktop computer in order to at least make anything work. It was buggy and slow. But I liked it as I got to learn it better. I'm excited to work with it more.

After all, my thoughts on the current laboratory work are:
It was a great lab because I always wanted to develop apps in the past, and here I got to do it under a deadline - no procrastination allowed.
It was truly an adventure, with ups and downs: I even had a system restore process during the project work because an external android emulator had got me a virus and crashed my system and the computer won't run! I was at the peak of insanity but happily my files were untouched and the project progress did not get affected.

After times like these I realize the person who said the following words was right:

"What doesn't kill you, makes you stronger!"

My repository: https://github.com/StasBizdiga/MIDPS

Table of contents