-finish documentation on installation, and getting data from HX711, and Challenges…

-in Main.xaml.cs: you don't call putRecordInDatabase everytime there's a network change. You just call GetLocalIP, but that doesn't update the database.

-Change return type of putRecordInDatabase from void to Task. And we should probably call it with await. see guidelines here:

<https://docs.microsoft.com/en-us/dotnet/csharp/async>

-in putRecordInDatabase: what should we do if there's an exception and we can't insert to the database ?

-If we choose to stay with the implementation by ip and sockets, then:

implementation of all the connect,send, read messages in the android app (and maybe in the Raspberry) should be changed a bit: you use timeout in sendScanned in order to wait for Receive function, but since in Receive you don't use await then it acts synchously and blocks until it returns. Also, in send you call async methods, but you don't await for them, so send also blocks until it returns. There are many solutions for these things. we need to decide if we want synchronious or asynchrounis I/O . If we use synchronious we need to set timeouts for each Read, write. The problem with synchronious is that when you give a false ip, then it tries to connect for a few minutes, and the UI of the android app isn't responsive to the user (Connect function blocks). So that's not good. If we'll use asynchorous sockets, then that won't happen.

However, maybe the android app can call a new UI thread that will keep spinning the progress bar.

some possible solutions depending on what we decide:

<https://docs.microsoft.com/en-us/dotnet/framework/network-programming/asynchronous-client-socket-example>

<https://docs.microsoft.com/en-us/dotnet/framework/network-programming/using-an-asynchronous-client-socket>

<https://docs.microsoft.com/en-us/dotnet/standard/asynchronous-programming-patterns/calling-synchronous-methods-asynchronously>

<https://stackoverflow.com/questions/1062035/how-to-configure-socket-connect-timeout>

also, we need to add exception handling to all I/O functions, and should we close the socket on the client side and the Raspberry's side ? we should do tests where guy A weighs himself and then guy B. Or guy A weighs himself and then weighs himself again after a minute. Right now we don't close on the client side, so this keeps the connection open with the Raspberry. We should decide if that's good.

-add ack messages between android and Rpi

-make sure you caught all exceptions in the RAspberry's program. The program should never crash except when there's no electricity to run it. So check in the documentation of all the functions you used, that you indeed caught all exceptions.

-we should now focus on tests that will try to crash the program. Like:

1. many people trying to start weight and connect to RPI

2. turn router off and on again

3. one guy weighs himself, and then another guy with a different phone

4. one guy weighs himself, and then after a minute weighs himself again.

## -make sure you make proper use of async and that your code doesn't block. there are several guidelines in this link in paragraph Important Info and Advice:

<https://docs.microsoft.com/en-us/dotnet/csharp/async>