Dynamic Personal Insurance

Final Report

4/25/2022

Author: Ignas Rocas, C00135830

Course: Software Development 4th year Project

Supervisor: Dr Greg Doyle

Contents

Introduction 2
Problems encountered & resolved 2
At the start of the project 2
In the middle of the project 2
At the end of the project 3
Achieved 3
Have not achieved 4
Learned 4
Do differently 5
Possible improvements & expansions 5
Users testing survey 5
Conclusion 6
References 6

Introduction

The dynamic personal insurance app aims to prove that insurance can be done differently. The project can be used as traditional Health insurance or by using Android watch technology with sensor data to dynamically change the price customer pays. The project also includes the client as a user, so it can manage the customer claims/details etc.. and review their progress.

This document describes the biggest difficulties that have been encountered and their resolve, the completed features of the project and how it could be improved more. Additionally, it lists all the things that have been learned and what could be done differently if what is known now.

Problems encountered & resolved

At the start of the project

- Research similar apps
  When researching similar applications I have discovered that there are not a lot of applications for the project to compare to. And the ones that are similar in a way, would not allow non-register customers to access. To solve the analysis problem, Customer reviews were looked at on the Android/Apple store to view the good & the bad of the applications.

- Technical difficulties
  The IDE (Visual Studio) was giving me a lot of errors, very slow, problems with downloading parts of the cross-platform parts, etc.
  Switching IDEs (Rider) eventually solved the problem.

- A lot of unknowns at one time.
  When starting the project there were a lot of unknowns at one time (c#, Xamarin, watch technologies, the connection between watch, Cloud mongo Db, MVVM in c#, etc.) and I found myself a bit lost. But eventually, breaking down the problem(project) into multiple parts started getting things done.

- The connection between mobile & watch.
  I have spent a long time looking for a way to connect the mobile & the watch so they can pass information between them. I have tried Google API such as Client Channel, and Message Client but I discovered that it won't work with the IOS side, since it needs an Android store. I have resolved the connection problem by using Bluetooth Low Latency (BLE) connection.

- Quote prediction using machine learning
  To predict a quote for the project, past customer data was needed. However, no health insurance data could be found anywhere online. To resolve the problem, the top Irish insurance companies such as VHI, Irish Life & Laya were investigated and the average quote data per customer of its age has been generated.

In the middle of the project

- Classification of the accelerometer data. (x, y, z)
  I have spent a long time to find a way to classify the steps (using machine learning) but I have found that the difficulty of completing that would be a 4th-year project on its own and

we have a limited time frame. To solve the problem I have ended up using a pre-created pedometer by "Pillai S Anu". [PIL17]

- Saving the step data to Mongo database & set up.
  When I was trying to save information to the Realm/Mongo database I knew I do asynchronously because the front-end of mobile would not cope, therefore, but I have been stuck with a pile of errors. After some research, I have discovered that there are a few rules when dealing with Realm which include "Don't allow passing live objects between threads". Therefore, to solve the problem I have created a realm instance for each time when data is to be saved and disposed of right after to avoid managed object clash between threads.[MON21]

- Sending information between the mobile & the watch.
  While I have solved the connection problem as discussed previously, I have discovered that since the biometric (x,y,z) data were gathered at an alarming rate (x1000 per min). Even though the BLE connection was used which main advantage is the minimal battery use, the 10% of the battery simply disappear in less than half an hour.
  To solve the problem, I have moved the classification of the steps on the watch and saved them cloud database while observing the database via mobile device.

- RealmDb logout bug
  I have a small problem but it was bothering me. I could not log out the user in case they wish to switch from client to customer. It gave me an existing user error till the app was closed. Finally, I discovered by research that the best way to keep track of realm database instances is to dispose of them as moving between pages.

- Implementation of AWS custom API(stateless).
  It took me a while to discover how/where to implement an API with a machine learning model, that would be extendable and use python language. I tried multiple different service providers such as Azure, Google, and Heroku but there is not a lot of information to fit my particular use case. (and simple) Eventually, I found a way to implement a stateless API using AWS and Docker, with the help of "Tanuj Jain" and his blog post. [JAI19]
  Additionally, since the AWS is expensive, I have copied/used the API on "pythonanywhere" since it is free.

At the end of the project

- Could not find other than my self-user that would test my project & possess a watch.

Achieved

The inputs are validated in real-time with the help of Xamarin-toolkit. (The text stays red till the right input is entered & if submits before then, an error message is combined from the data)

The main function of the project achieved:

- Customer-side
  - Register/setup functionality
    - Get a quote. (Using stateless custom API)
    - Confirm email. (Using stateless custom API)

        - Record personal details in the registration phase. (Records customer/Policy)
        - Identify the type of user card entered while helping the user.[MEH20]
        - Process payment safely and securely with Stripe. (Without keeping any information)
    - Log in as a customer.
    - Connect & Record steps via the android watch. (Using BLE)
    - View their steps are taken (progress) in real-time. (minimal time frame)
    - Record steps without the need for the phone after initialization. (For a limited time)
    - Record a claim/View previous claims.
    - View & Change profile details.
    - View & Change policy(every 3 months, using stateless custom API)
    - View daily & weekly step report in a form of line charts.
    - Use the rewards when the customer pays.
    - Recover old account if a customer comes back after 2 months.
    - Change password
    - Log-out.
- Client-side
    - Register. (Record account details)
    - View all customers.
    - Act as any customer. (Create claim, change policy etc…)
    - View & resolve customer claims with an automatic email notification sent.
    - View previous claims.
    - View & allow/deny customers policy updates with an automatic email notification sent.
    - View previous policies.
    - Reset the customer's password.

Have not achieved

Since there is a time constraint, these functionalities were not implemented:

- Change/reset the password for the client.
- Extra report details. (Average steps per week, month)
- A better way to classify steps.
- Using other forms of sensors.
- Record sleeping habits.

Learned

As previously mentioned, I never learned so much from a project. Some of the things include:

- C# programming language
- To use Xamarin forms framework
- Implementation of "Model-View-ViewModel" (MVVM) design pattern in c#.
- Xamarin forms libraries, such as the Xamarin toolkit, its essentials, Bluetooth 3rd party etc.…
- How the BLE works.
- How to use Docker
- Implement Stateless API with AWS EC2 & Docker
- Sending emails using the SMTP library

- How to implement Stripe & how it works.
- How to create an app for an Android watch. (C#/Xamarin)
- How to encode/send/receive/decode HTTP requests with Xamarin/python.
- Using Mongo Cloud database with .Net in conjunction with the Realm cloud app.
- Using Realm cloud app Functions.
- How to use JSON.
- Using basic machine learning modelling.

Do differently

- I have spent a lot of time trying to find a way of classifying the steps while using machine learning. If I knew that now, I would spend less time focusing on machine learning and more time on a different part of the project.
- Use the IOS apple watch since it is more popular in the market, but I have purchased an Android watch therefore I had no more choice.
- I should have looked for users that possess an android watch for the testing sooner since it is more difficult to come by in comparison to an android phone.

Possible improvements & expansions

1. Use external API for steps classification, such as Google Fit.
2. Implement the use of different types of sensors that are available by the watch.
3. Customer sleeping patterns.
4. Daily/Weekly customer surveys. (Where they fill an app form about how they feel.)
5. Include sleep tracking.
6. Feature to add family members to the policy.
7. Group health insurance. (Introduce Gamification as part of it)
8. Extend to life insurance.
9. Extend to the car insurance.
10. Extend to the house insurance.

Users testing survey

Know friends and relatives have been given an android part of the project to try out and a survey to fill in about their experience.

	How old are you? <24 / 25-36 37-45 / 45+	How long did it take for you to sign up? 3min/5min /10min	Was anything confusing?	Please rate the User interface from 1 to 10. (Colors, pictures navigation, etc..)	Any additional comments?
Geraldine	45+	5	I was not sure about the Quote options.	8	Would be nice to get a Quote on the computer maybe?
Dominic	25-36	3	No	8	I think it looks good.
Giedre	45+	10	Seemed to be clear	7	None

Daithi	<24	3	Not sure why can't I record steps if I don't own a watch.	6	I think the app should be more colourful.

Conclusion

I am pleased with how the project has turned out; it was a thoroughly great experience of how to solve problems when they have been encountered, working under pressure while following deadlines. I have learned a lot about mobile development, machine learning & use of API.
The most important thing that the project thought me if can't figure something out for a while, take a break and try again or if can't figure it out for a day talk to someone about it. (It will open the bubble)

References

[PIL17] Pillai S Anu, (2017), "*Create a Simple Pedometer and Step Counter in Android*", http://www.gadgetsaint.com/android/create-pedometer-step-counter-android/, Last Accessed: 16/04/2022

[MON21] Mongodb, (2022), "Threading - .NET SDK", https://www.mongodb.com/docs/realm/sdk/dotnet/advanced-guides/threading/, Last Accessed: 16/04/2022

[JAI19] Jain T., (2019), "Simple way to deploy machine learning models to cloud", https://towardsdatascience.com/simple-way-to-deploy-machine-learning-models-to-cloud-fd58b771fdcf, Last Accessed: 16/04/2022

[MEH20] Mehers D., "A Xamarin Stripe example", https://damian.fyi/xamarin/2020/08/07/xamarin-stripe.html, Last Accessed: 16/04/2022