Dynamic Personal Insurance

Functional Specification

4/11/2022

Author: Ignas Rocas, C00135830

Course: Software Development 4th year Project

Supervisor: Dr Greg Doyle

Abstract

The aim of this project (“Dynamic personalised insurance product application”) is to develop an app that can run on multiple platforms, such as Android, IOS and Wear OS while involving two sides of the application.

Firstly, the customer can manipulate and observe insurance policy costs with lifestyle actions involving the watch which will facilitate the gathering of the lifestyle data.

Secondly, the Client section can manage customers & their policies while observing their lifestyle action data.

Table of Contents

Application Definition

The purpose of the project is to produce an insurance cross-platform application for a new type of insurance, Dynamic personalized insurance. The application allows managing users that purchase insurance while modifying its policy price with the use of tracking movement data from the smartwatch.

The application is separated into three sides:

- The User-side
- The Customer-side
- The Client-side

The user has to sign up and pay for traditional insurance to become a customer, where they can utilize the functionality of the application.
The customer then can have access to a process of connecting the watch, which involves downloading and installing a companion app to the watch.
When the watch is connected to the application the authorized customer may switch on the monitoring of the accelerometer sensor which is located on their watch.
The mobile application sends the details via Bluetooth to the watch so it can connect to Cloud DB and store monitored and classified steps via a built-in classification algorithm. Finally, the mobile application monitors the cloud database and displays the steps in real-time.
The end goal of the customer is to receive as much cash back as possible on the next month's policy price. The end goal of the client is to review/resolve and assist customer claims, policy updates etc…

Project features

The project features identified are:

Core functionalities (Customer)

1. Handheld device connection and authorization process to wearable.
   Which includes biometric data collection/process and storage.
2. The client reviews customers' life personal data
3. Prescribe a Quote for an insurance policy (using Custom API).
4. Creating policy.
5. Create/View Reward.
6. Registration.
7. Login.

Non-Core functionalities: (Customer/Client)

1. Update/Review policy (Customer)
2. Create/Update claim (Customer)
3. Update personal data (Customer)
4. Client Registration.
5. Client Log in.
6. Client review customers.
7. Client Resolve Claim.
8. Modification of policy (Client).
9. Allow policy updates. (Client)
10. Change customer profile details. (Client)
11. Manage selected customer Claims. (Client)

12. View Previous policy updates. (Client)
13. Change password. (Customer)
14. Reset selected customer password. (Client)
15. View Report (Customer/Client)
16. Pay (payment system while using rewards)
17. Customer change/reset password.
18. Send customer email notification. (When client resolved claim or allows customer policy update)
19. Confirm email at registration.
20. Log out.

User Groups

The project contains two main user groups, Customer and Client. Another type of user is a "User" which is a customer (unpaid customer). The application is most of the time used by the Customer while recording/gathering in comparison to the Client. The Client can do most of the things that the customer can do in the app.

Customer

It's a user that is authorized (registration completed) to access the application. The customer is happy to pay for the traditional premium but looking to lower its cost by use of technology.

The minimum age of the Customer is 18 since most of the insurance products in Ireland can't be bought below that age. To become a customer below 18, a person has to be added to over 18's insurance but that would extend to Family insurance. Additionally, the maximum age is 64 as research provided as after 65 customers may due to unpredictable risk factors and involves a different type of insurance "senior citizens insurance".

[HDF21]

Other age groups in between are 18-20, 21-25, and 26-35. The difference between them is the premium price.

However, the project involves new wearable technologies most likely the project will be used by younger person groups and not older ones.

Client

It is a user that has been authorized by its company (needs a special registration code) to access the app. Clients are personnel that work at the insurance providers and they are reasonable in managing the customer's updated policies, open claims and sell policies (salesmen and/or support). They can do everything the customer can do except create biometric data or rewards.

User

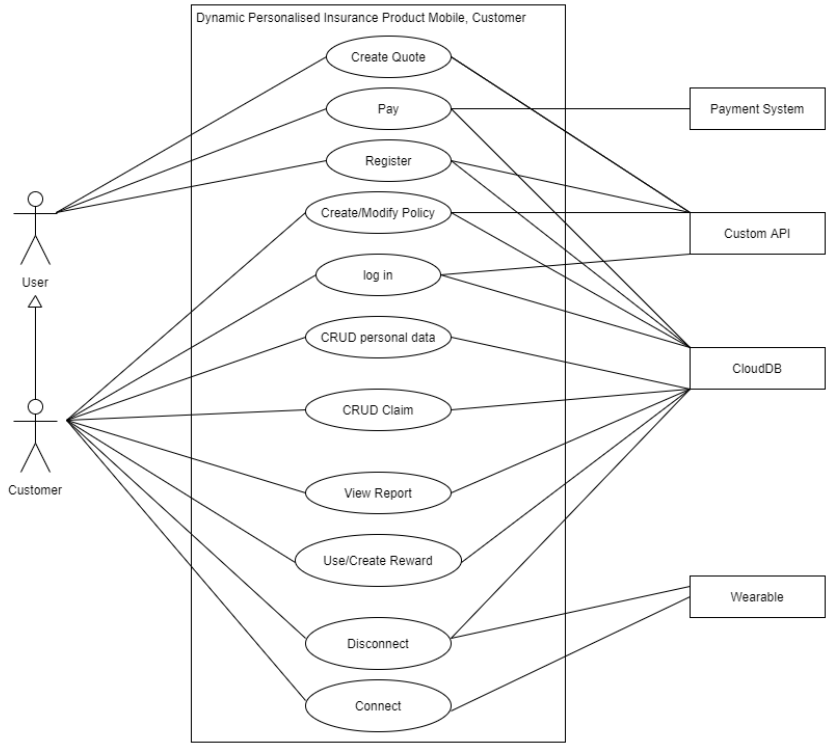
The user is an unauthorized customer to access the app after its registration. In the case of the customer:

- Registers and does not pay.
- The policy expires.

At these times the customer must process payment to access the app.

Use Cases

Use Case Diagrams



1 Figure, Customer Use Case Diagram

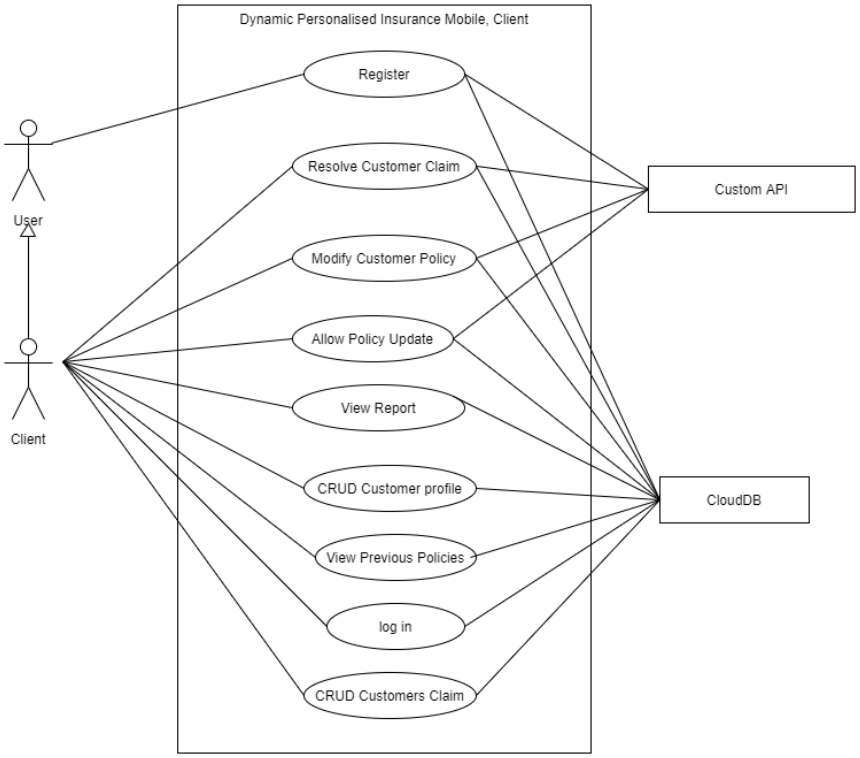


Figure 2, Client Use Case Diagram

Brief Use Cases

Customer brief use cases

Name:	Create Quote
Actors:	User, Custom API, App
Description:	The use case begins when the user wishes to become a customer of dynamic insurance, while the application has been installed and opened. On the Login screen, the user selects the Get Quote option which navigates to the Quote page and lets the user pick options available for the policy including their age. Details selected are authorized by the app and using the Custom API classified in order to get the price, which is displayed on the user's screen with an option to accept it or cancel it. (Accepting redirects the user to the registration screen.) The use case ends when the user accepts the price and gets directed to the registration screen.

Name:	Register
Actors:	User, Cloud DB & App
Description:	The use case begins when the user wishes to become a customer of dynamic insurance, while the Quote has been issued and the user has accepted the price. On the Registration screen, options are provided for the user to enter their personal information such as email, password, name and other The email has been entered and confirmed. (See Confirm email use case) The address has been validated. (See Validate address use case) The app validates the input, registers a customer, sets a Policy for them and saves info to Cloud DB which results in transferring the user to the payment page. The use case ends when the user has been transferred to the payment page.

Name:	Validate address
Actors:	User, App
Description:	The use case begins when the user wishes to confirm their address. The app is installed and opened and the user has been navigated to the address pop-up. The user enters their details such as street, country, postcode and others. The app validates inputs as they are being provided. The selected options to return submit the info and the app navigates them back to the registration page. The use case ends when the user leaves the address pop up.

Name:	Confirm email
Actors:	User, Custom API, App
Description:	The use case begins when the user wishes to confirm the email. The app has been installed and opened and the user currently is on the registration page. As the user enters the correct email, the option to confirm the email appears. The user selects the option to confirm email, which makes the app generate a random sequence of characters, send them to the user's email via Custom API and provide an option to enter the generated code. The user checks their email and enters the code provided which shows that the email is confirmed visually. The use case ends when the customer provides the correct email confirmation code.

Name:	Pay
Actors:	User, App, Payment System, Cloud DB
Description:	The use case begins when the user wishes to become a customer of dynamic insurance. The app has been installed, is open and the user is currently located on the payment page. The App provides users options for the user to enter their card details (card number, Secret Code Expiry date, postcode) while displaying the policy price which they are about to pay for. As details are inserted by the user, the inputs are validated which app shows it visually. (If the user is a pre-existing customer the option to use rewards will appear) After valid details are inserted by the user, the app processes the payment using the Payment System. After the process is completed, the user is notified and if successful redirected to the login page. The use case ends when the user is transferred back to the login page.

Name:	Create/Modify Policy or Modify customers policy (Client)
Actors:	App, Email, Customer, Cloud DB, Custom API
Description:	The use case begins when a customer wishes to update an insurance policy. The app is installed, open and the policy page has been opened. The app gets the existing policy from Cloud DB and displays it to the user. If it is not updated recently, the customer goes through the process of getting a new Quote. (Please see Create Quote use case). After the process is completed, the app saves a new policy in Cloud DB and does not allow the policy updated for a specific period of time. The use case ends when a customer leaves the policy page.

Name:	Log in (Customer/Client)
Actors:	The app, User, Cloud DB
Description:	The use case begins when a user wishes to enter the system and the app is opened. At the login screen, a user enters their credentials. The app validates the inputs and checks which type of user is trying to log in with the help of Cloud DB. The use case ends when a user leaves the login screen.

Name:	CRUD Personal Data
Actors:	Customer, App, Cloud DB
Description:	The use case begins when the customer wishes to read or change the personal details. The app is installed and the personal details page is open. The customer changes personal details and the app validates them as they are being changed. The customer selects the update details option which makes the app to updated details on the Cloud DB and notifies the customer about the successful process. The use case ends when a customer leaves the personal details screen.

Name:	CRUD Claim (Customer) / CRUD Customers Claim (Client)
Actors:	Customer, Cloud DB, App
Description:	The use case begins when the customer wishes to create a new claim or view an existing not resolved claim. The app is installed and opened on the Claim page. The customer enters details for the claim to be reviewed, such as hospital code, patient number and/or extra information. The app validates the provided info and updates Cloud DB. The use case ends when a customer leaves the claims screen.

Name:	Connect
Actors:	Customer, Wearable, Cloud DB, App
Description:	The use case begins when a customer wishes to start monitoring movement data via the Wearable device. The mobile app and wearable app are installed and opened. The customer has logged in and opened the home page. The customer toggles switch to start movement data monitoring. The app makes sure the Bluetooth is on and connects to the wearable(app). The app starts to listen for new entry inserts in Cloud DB while updating the UI. The use case ends when a customer leaves the home page or turns off the app. (stops listening for new entries insets in Cloud DB)

Name:	Disconnect
Actors:	Customer, Wearable, Cloud DB, App
Description:	The use case begins when a customer wishes to stop monitoring movement data via the Wearable device. The mobile app and wearable app are installed and opened. The customer has logged in and opened the home page. The customer toggles switch to stop movement data monitoring. The app updates the Cloud DB switch and stops listening for new entries to Cloud DB. The use case ends when the customer toggles the switch to turn off monitoring.

Name:	Create/Use Reward
Actors:	Customer, Wearable, Cloud DB, App
Description:	The use case begins when: • A customer wishes to Use their reward. The customer is located on the payment page and selects earned rewards. The app will re-calculate/display price change. The rewards are removed as the payment process is completed. • The customer logged in for the first time. The use case ends when the reward is created or removed.

Name:	View Report
Actors:	Customer, Cloud DB, App
Description:	The use case begins when a customer wishes to view a report of steps that have been taken past 7 days/weeks. The app is installed, opened, and the customer navigated to the reports page. The app compiles the report using retrieved Cloud DB biometric data into weekly/daily charts and displays it to the customer. (If any has been submitted) The use case ends when a customer leaves the report page.

Client brief use cases

Name:	Resolve Customer Claim
Actors:	Client, Cloud DB, Custom API
Description:	The use case begins when a client wishes to view and/or resolve a claim. The app is installed and opened and the client is located at the selected customer claim. The App displays option to CRUD claim and displays opened claim if it exists. The client selects the option to resolve/deny the claim. If deny option is chosen the app will ask for the reason and submits the resolve with the use of Cloud DB. The app notifies the customer of the claims to change via Custom API(Email). The use case ends when a customer leaves the claims screen.

Name:	Allow Policy Update
Actors:	App, Client, Cloud DB, Custom API
Description:	The use case begins when a client wishes to manage customer policy updates. The app has been installed, turned on and the client navigated to the updated policy page. The app displays updated policies from all customers. The client selects a particular policy they wish to change, and the app redirects the policy page while displaying customer selected options retrieved from Cloud DB. The client reviews the existing policy and chooses to allow/deny the policy update option provided. The app sends the customer a notification email via Custom API and updates the Cloud DB. The use case ends when a user leaves the policy page.

Name:	Register (Client)
Actors:	The app, User, Cloud DB, Custom API
Description:	The use case begins when a user wishes to enter the system. The app is installed, is open and the user is located on the login screen. The user selects the option for client registration. The app redirects to the client registration page and displays options for details to be provided such as name, email, password, security code etc… The user enters the details required, and the app validates to its standards and checks the security code via Custom API. The use case ends when a user leaves the registration screen.

Name:	View Report
Actors:	Client, App, Cloud DB
Description:	The use case begins when a client wishes to view Report. The application is installed, turned on and the client is navigated to the report page. The app retrieves biometric data from the Cloud DB, compiles it to weekly specifics and displays it to them. The use case ends when a client leaves the report page.

Name:	CRUD Customer profile
Actors:	Client, Cloud DB, App
Description:	The use case begins when a client wishes to change/view the personal profile of a specific customer. The app has been installed, turned on and the client has navigated to the selected customer. The app displays existing customer profile details. The client changes the details and selects the option to change them. (not including password or email) The app updates details using Cloud DB. The use case ends when a client leaves the customer's profile management facilities.

Name:	View Previous Policies
Actors:	Client, Cloud DB, App
Description:	The use case begins when a client wishes to change/view the personal profile of a specific customer. The app has been installed, on and the client has navigated to selected customer policy management facilities where the app displays option to view previous policies. The client selects the option provided and the app displays previous policies that have been retrieved via Cloud DB. The use case ends when a client leaves the customer policy management facilities.

Detail Use Cases

Customer detail use cases

Detail use Case: Create Quote

Actor(s): User, Custom API, App

Preconditions: The app is installed, opened and the get quote option is selected at the login screen.

Main scenario

1. The app provides facilities for the user to pick the insurance options to get an insurance quote. (Age, Hospitals, Hospital Excess, Cover, Plan, Smoker)
2. The user selects from the options provided.
3. The app validates options chosen by the user.
4. The app sends cover data to API.
5. API predicts the price using a pre-trained Machine learning model and sends back the price.
6. The app receives a response with the price.
7. The app displays a pop up with the price, including the option to accept or decline.
8. The user accepts the price.
9. The app redirects to the registration screen.

Alternatives

4a. There is no internet connection.

1) The app displays an error message.
2) The user fixes the connectivity issue.
3) Back to main scenario 4.

4b. It takes over 2min for the custom API to respond.

1. The app displays an error message while asking the user to try again later.
2. Back to main scenario 4.

7a. The user declines the price issued. (It can happen multiple times)

1) The app closes the price pop up leaving the chosen option.
2) The user changes the insurance options.
3) Back to main scenario 2.

Detail use Case: Register

Actor(s): User, Cloud DB & App

Preconditions: The app is installed, turned on and the user is on the registration page.

Main scenario

1. The app provides facilities for the user to enter their details. (Name, Age, username, email, password etc.)
2. The user enters the personal information.
3. The app validates the information provided.
4. The App creates a Customer and policy using the provided info and the Quote.
5. The app updates details on Cloud DB.
6. The app displays a success notification.
7. Navigates to the payment page.

Alternatives

3a. The information provided incorrect type.

1. The app displays the incorrect type of details provided.
2. The user re-enters the details needed and selects the option to complete.
3. Continue to the main scenario 2.

4a. The user does not have a network connection.

1. The app displays the error message
2. The user fixes the network connection issue.
3. Continue to the main scenario 2.

Detail use Case: Validate address

Actor(s): User & the App

Preconditions: The app is installed, turned on and the user is on the address page. (Transferred via registration)

Main scenario

1. The app provides facilities for the user to enter their address details. (House number, street, postcode etc…)
2. The user enters the personal information.
3. The app validates the information provided.
4. The app navigates back to the registration page.

Alternatives

3a. The information provided incorrect type.

4. The app displays the incorrect type of details provided.
5. The user re-enters the details needed and selects the option to complete.
6. Continue to the main scenario 2.

Detail use Case: Confirm email

Actor(s): User, Custom API, App

Preconditions: The app is installed, turned on and the user is on the registration page.

Main scenario

1. The app provides facilities for the user to enter their email address.
2. The user enters the personal information.
3. The app validates the information provided and displays confirm email option.
4. The user selects the option.
5. The app generates a random sequence of characters (code).
6. The app sends an email via custom API with the generated code.
7. The app displays option for the user to enter the email code.
8. The user enters provided code.
9. The app validates the code and displays a confirmation message.
10. The app disables the option to change the email address.
11. The app continues with registration.

Alternatives

9a. The information provided is incorrect.

1. The app displays an error message.
2. Back to main scenario 8.

9a. The information provided is incorrect.

1. The app asks gives the option to resend the confirmation code
2. The user chooses that option.
3. Back to main scenario 5.

9a. The information provided is incorrect.

1. The app displays an error message.
2. The app asks gives the option to try again later.
3. The user chooses the option.
4. Back to main scenario 3.

Detail use Case: Pay

Actor(s): User, App, Payment System, Cloud DB

Preconditions: The app is installed, opened and the user navigated to the payment page.

Main scenario

1. The app provides an option to enter customer details and displays price/ postcode.
2. The user enters card details such as card number, security code and expiry date.
3. The user selects the option to submit the details.
4. The app validates the user details.
5. The app uses a payment system to process the payment.
6. The payment system validates details.
7. The payment system processes the payment.
8. The payment system confirms that the payment has been processed successfully.

9. The app displays a successful notification.
10. The app navigates to the login page.

Alternatives

3a. The user does not have an internet connection.
    1. The app displays an error message.
    2. The user fixes the connectivity issue.
    3. Back to main scenario 2.

4a. The details provided are not the correct type.
    1. The app displays an error message.
    2. Back to main scenario 2.

6a. The card details are not valid.
    1. The payment system returns an error.
    2. The app displays an error message.
    3. Back to main scenario 2.

Detail use Case: Log in (Customer/Client)

Actor(s): App, User Cloud DB

Preconditions: The app is installed and opened.

Main scenario
1. The app provides an option to enter users' credentials.
2. The user enters their credentials.
3. The user selects the option to log in.
4. The app validates the credentials using Cloud DB.
5. The app checks which type of user it is using Cloud DB.
6. The app navigates to the app.

3a. The user does not have an internet connection
    1. The app displays an error message
    2. Back to main scenario 2.

4a. The credentials are invalid.
    1. The app displays an error message
    2. Back to main scenario 2.

6a. The user is a type of normal customer.
    1. The app navigates to the customer type of navigational app.
6b. The user is type unpaid customer.
    1. The app navigates to the payment page.
6c. The user is a typical client.
    2. The app navigates to the client type of navigational app.

Detail use Case: CRUD Personal Data

Actor(s): Customer, App, Cloud DB

Preconditions: The app is installed, opened and the personal details page is open.

Main scenario

1. The app displays existing information that is retrieved from Cloud DB.
2. The customer changes the information.
3. The customer selects submit changes option.
4. The app validates the new information as its being provided.
5. The app sends the details to be saved on Cloud DB.
6. The Cloud DB updates the detail provided.
7. The app displays a successful message.

Alternatives

3a. The device does not have an internet connection.

1. The app displays an error message.
2. The customer fixes the issue.
3. Back to main scenario 2.

4a. The credentials provided are the wrong type.

1. The app displays the error message.
2. Back to main scenario 2.

Detail use Case: CRUD Claim (Customer) / CRUD Customers Claim (Client)

Actor(s): Customer, Cloud DB, App

Preconditions: The app is installed and opened on the Claim page.

Main scenario

1. The app does not find the existing claim by Cloud DB.
2. The app provided empty facilities for the data to be provided.
3. The customer enters details such as hospital code, patient number and extra info.
4. The customer submits details.
5. The app validates the details provided.
6. The app saves data to Cloud DB.
7. The app displays a success message.

Alternatives

1a. The app finds an existing claim via Cloud DB.

1. The app retrieves and displays the existing (open) Claim to the customer and disables the editing.

5a. The details provided are the wrong type.

1. The app displays an error message with the wrong type of data provided.
2. Back to main scenario 4.

5b. The devices do not have a network connection.

1. The displays no internet connection error message.
2. Back to main scenario 4.

Detail use Case: Connect

Actor(s): Customer, Wearable, Cloud DB, App

Preconditions: The mobile app and wearable app are installed and opened. The Customer has navigated to the home page.

Main scenario

1. The app gets a monitoring switch(off) from Cloud DB.
2. The customer toggles the switch.
3. The app turns on the switch visually.
4. The app finds connected to Bluetooth le know devices and the application.
5. The app sends a message with an email and password to the Wearable.
6. The app starts listening to the Cloud database for new Mov data added.

Alternatives

1a. The app gets a monitoring switch(on) from Cloud DB.

1. Back to the main scenario 3.

4a. The device does have a Bluetooth connection.

1. The app displays an error message.
2. Back to main scenario 3.

4a. The device does not find the service which the Wearable app creates.

1. The app displays a wearable app not found message.
2. Back to main scenario 3.

4a. The device does not find any Bluetooth LE device connected.

1. The app displays a wearable app not found message.
2. Back to main scenario 3.

Detail use Case: Create/Use Reward

Actor(s): Customer, Wearable, Cloud DB, App

Preconditions: The app is installed, opened and the customer is navigated to the payment page.

Main scenario

1. The App gets rewards completed from Cloud App.
2. The app calculates the total price that can be deducted from the total price.
3. The app displays the price.
4. The customer select's option to deduct the reward from played price.
5. The app changes the displayed price.
6. The customer selected to proceed with payment.
7. The use case ends and for the following action please see the "Pay" use case.

Alternatives

1a. The app does not find any rewards completed.

1. Return to main scenario 7.

Detail use Case: View Report (Customer)

Actor(s): Customer, Cloud DB, App

Preconditions: The app is installed, opened, and the customer navigated to the reports page.

Main scenario

1. The App gets biometric data from the Cloud DB.
2. The app compiles daily and weekly charts.
3. The app displays the charts to the customer.

Alternatives

1a. The app does not find any biometric data from the Cloud DB.

    1. The app displays a notification to the customer instead of a chart.

1b. The app only finds less than a week's biometric data from Cloud DB.

    1. The app displays a daily chart only.

Client detail use cases

Detail use Case: View/Customer Resolve Claim

Actor(s): Client, Cloud DB, Custom API

Preconditions: The app is installed, opened and the client is located at the selected customer claim.

Main scenario

1. The app retrieves the existing open selected customer claim.
2. The app displays existing open claims details.
3. The client selects the option to resolve (accepts) the claim.
4. The app asks for confirmation.
5. The client confirms.
6. The app updates Cloud DB.
7. The app sends an email via Custom API to notify a customer about changes.
8. The use case ends. (For the following action, please see CRUD Claim (Customer) / CRUD Customers Claim (Client) use case).

Alternatives

1a. The app does not find any open customer claims.

    1. Back to main scenario 8.

4a. The device does not have internet connectivity.

    1. The app displays a connectivity error.
    2. The client resolves the issue.
    3. Return to main scenario 2.

3a. The client selects the option to resolve the claim by denying it.

1. The app asks for confirmation.
2. The client confirms the process.

3. The app asks for a reason.
4. The client enters the reason for denying the claim.
5. Return to main scenario 6.

3b. The client selects the option to resolve the claim by denying it.

1. The app asks for confirmation.
2. The client confirms the process.
3. The app asks for a reason.
4. The client does not enter the reason.
5. The app displays an error message.
6. Back to main scenario 2.

Detail use Case: Allow Policy Update

Actor(s): App, Client, Cloud DB, Custom API

Preconditions: The app has been installed, turned on and the client navigated to the updated policy page.

Main scenario

1. The app retrieves all open policies from Cloud DB.
2. The app displays the policies found.
3. The client selects the policy they what to update.
4. The app navigates to the selected policy page.
5. The app navigates to the policy page.
6. The app displays policy to be revised and an option for them to allow the update.
7. The client selects to option to resolve the update.
8. The app asks for confirmation.
9. The client gives the confirmation.
10. The app updates the Cloud DB and sends emails about the action to the customer via a custom API.
11. The app displays a success message.

Alternatives

2a. The app does not find any updated policies.

1. The app shows that no policies were updated yet instead of the policy list.

7a. The device does not internet connection.

1. The app displays a network connection issue message.
2. The client fixes the issue.
3. Back to main scenario 7.

Detail use Case: Register (Client)

Actor(s): App, User, Cloud DB, Custom API

Preconditions: The app is installed, opened and the user is located at the client registration screen.

Main scenario

1. The user enters details into provided fields. (Name, email, security code, password, etc...)
2. The user selects the option to submit details.
3. The app validates the security code via Custom API.
4. The app validates other information and updates Cloud DB.
5. The app displays a success message and redirects to the login page.

Alternatives

3a. The Custom API does not validate the correct security code.

1. The app displays an error message.
2. Back to main scenario 1.

3b. The device does not have an internet connection.

1. The app displays a connectivity error message.
2. The user fixes the network issue.
3. Back to main scenario 2.

4a. Other info is an invalid type.

1. The app displays an invalid info error message.
2. Back to main scenario 1.

Detail use Case: View Report (Client)

Actor(s): Client, Cloud DB, App

Preconditions: The app is installed, opened, and the Client navigated to the reports page.

Main scenario

4. The App gets all biometric data from the Cloud DB.
5. The app compiles weekly charts.
6. The app displays the charts to the Client.

Alternatives

1a. The app does not find any biometric data from the Cloud DB.

2. The app displays a notification to the Client instead of a chart.

Metrics (FURPS+)

The Supplementary Specification describes quantifiable non-functional requirements as follows:

Functionality

- System errors - Errors should be saved into a cloud therefore examination can be made.

Usability

- A user must become a customer within 5 min if they possess all information needed, such as card number, email address etc…
- The customer should be able to view their data within a min from the login screen.

Reliability

- Connect the wearable and handheld should be successful 90% of the time.
- The app should not crash 99% of the time.

Performance

- The application should load within 2 seconds.
- The application should shut down within a second.
- Switching between management facilities should take only a spit of a sec under a stressed system.

Supportability

- Installation after download should not take longer than 2 min.(dependent on the device)

+

- The implementation language should be C# (Xamarin Forms).
- The interface should be colourful and clean.
- The system should be extendable to implement different insurance products

Inspiration

The Dynamic insurance product is a type of insurance that changes dynamically depending on customer lifestyle behaviour/choices and awards them.

The dynamic part of insurance is a very new topic since the number of wearable devices worldwide is increasing very quickly (more than 5 times from 2014 to 2019). Another block that prevents the incorporation of the new insurance model is people since it is difficult to trust the technology.

They are quite a few insurance companies that try to incorporate a new Dynamic model into their insurance but when they do, most of them don't follow it successfully. Some of these companies include:

- *MyLife app* by *Irish life* [MYL21]
- *Activ Health app* by *Aditya Birla Capital* [HEA19]
- *Discovery app by Discovery* [DIS21]

Several other companies are trying to implement a dynamic insurance model but they are focusing on private sector customers, such as companies with their employers. (*yulife*)
Only one out of three companies mentioned above fully incorporates the dynamic insurance model(which is, giving the customer cash-back) Activ Health.
The main difference between this project and the other applications is the data gathering/processing. All the applications reviewed does not process/ gather data produced by the wearable. Instead, they use several fit tracing applications to acquire already processed data.
[MON18]

References

[MYL21] MyLife by Irish Life, (06/07/2021) [Mobile app], Google Play Store,
https://play.google.com/store/apps/details?id=com.irishlife.mylife
Accessed: 25/10/2021

[HEA19] Activ Health, (27/09/2021) [Mobile app], Google Play Store,
https://play.google.com/store/apps/details?id=com.adityabirlahealth.insurance
Accessed: 25/10/2021

[DIS21] Discovery, (20/10/2021), [Mobile app], Apple Store,
https://apps.apple.com/za/app/discovery/id458077762?ls=1
Accessed: 25/10/2021

[MON18] MoneySuperMarket, (17/08/2018), By moneysupermarket.com,
https://www.moneysupermarket.com/life-insurance/personal-wearables/
Accessed: 24/11/2021

[HDF21] hdfcergo.com, (21/10/2021), "Is there an Age Limit for Buying a Health Insurance Policy?",
https://www.hdfcergo.com/blogs/health-insurance/age-limit-for-buying-health-insurance-policy,
Accessed: 02/01/2022