

Dynamic Personal Insurance

Design Manual

4/25/2022

Author: Ignas Rocas, C00135830 Course: Software Development 4th year Project Supervisor: Dr Greg Doyle

Abstract

The purpose of the project is to develop a cross-platform app and an android watch app. With their combined efforts, the project provides an ability for customers to purchase/manage Dynamic Personalized Insurance (DPI) while attached to a companion android watch. Furthermore, a client can view/manage their data/details that have been recorded by watch movement sensors.

Table of Contents

Abstract	1
Introduction	1
Technology Architecture	2
User interfaces	2
UI first draft	3
UI final product	3
Domain model	7
Class Diagram	8
System Sequence diagrams	9
Database Models/Schema	13
Cloud database schema	14
Project Plan	17

Introduction

The following document is used to document the Design of the DPI application. The Design involves a show of the overall architecture abstractly with major components interconnecting.

Furthermore, the document proceeds in describing the major components/connections as seen in architecture. The main components are Front-end, Software, and Back-End. Front-end detailed as UI design, Software explained as Domain model, Class Diagram and Sequence diagrams. In conjunction with the back end, the Database Models/Schema section is established.

Technology Architecture



Figure 1 - Technology architecture.

Figure 1 describes the Technology architecture of the project. A mobile app running Xamarin forms is the main staple/access point of the project. The user (Customer/Client) interacts with it to perform multiple CRUD operations.

The Xamarin mobile app interacts with :

- Custom API performs multiple functions, such as Inference quote price (Using Machine learning) for the customer to create a Policy, send emails (Via Simple Mail Transfer Protocol library) to the customer and manage client registration codes.
- Android watch App, via Bluetooth to initiate tracking of accelerometer sensor data identifying as a step, updating the Mongo DB and saves customers login details to the local SQL lite database.
- Mongo DB to Store/Manipulate data such as Customers, Clients, Policies, Rewards etc.

User interfaces

Figure 2 and 3 shows a prototype of screens the customer may see when they have logged in whereas Figure 4 & 5 display the management facilities that the client may use. By clicking on a customer in figure 4 a temporary screen will be opened where full customer profile details/policy maybe change. In Figure 5, a new window will be opened when the client selects one of the claims, where it can be solved.

UI first draft

	🗢 🖌 🖥 10	16:56			~
nic lı	nsurance		=	Dynamic Ir	surance
				2	
			Q	Profile	
Keep	o going !		335	Policy	
			Y	Rewards	
)		0	Claims	
Vext	Reward		Ŕ	Life style data	
otal Ea	rned: 50€		ō	Watch Connecti	ion
Custom	O □ Per Cash back scr	reen	Figure 3	⊲ - Custome	O er Navigat
Custom	O □ Per Cash back scr	reen 16:66	Figure 3	⊲ - Custome	O er Navigat.
1 Custom	O □ eer Cash back scr	reen 18:56	Figure 3		O rr Navigati ♀ ⊿ urance
Custom ynamic Ir	O □ eer Cash back scr nsurance Cams	reen 18:50	Figure 3	 Custome Oynamic Insu stomers 	O Pr Navigati Urance
ustom namic II	Cash back scr Cash back scr Cams arch here Q	reen 16:56	Figure 3	Custome - Custome Dynamic Insu	O er Navigati urance eaims O
namic II ners Sei	Cash back scr nsurance arch here	reen 16:56	Figure 3	Custome - Custome Oynamic Insu atomers Unsolved Cla All Claims	O er Navigati urance etams aims O o
namic li ners Sei	O □ Peer Cash back scr Insurance Claims arch here Q John 123451231231 Marry	16:50	Figure 3	C - Customer Dynamic Insu Moners Unsolved Cla All Claims omer Name :	O er Navigati urance aims O o John 10/02/2021
rnamic Ir mers sei umber :	O □ Per Cash back scr Insurance Clams arch here Q John 123451231231 Marry 23212312123	reen 16:56	Figure 3	Customer Dynamic Insu Stomers Unsolved Cla All Claims omer Name :	O or Navigat: urance aims O Sohn 10/02/2021 Marry 10/02/2021
ynamic I ynamic I yners sei umber : umber :	Image: Cash back scr Image: Cash back scr	reen 16:56	Figure 3	Customer Customer Oynamic Insu Monor Insu Marcians All Claims omer Name : omer Name : omer Name :	O Pr Navigat: Urance aims O S John 10/02/2021 Jelius 10/02/2021
Custom ynamic I tomers ser sumber : i umber : i umber : i umber :	Image: Cash back scr Image: Cash back scr	16:56	Figure 3	Customer Customer Stomers Unsolved Cla All Claims omer Name : omer Name : omer Name : omer Name :	O Pr Navigat Urance aims O O John 10/02/2021 Marry 10/02/2021 Bobo 10/02/2021
] Custom Pynamic II Pynamic II P	Image: Cash back scr Image: Cash back scr	reen 16:56	Figure 3	Customer Customer Stomers Unsolved Cla All Claims omer Name : omer Name : omer Name : omer Name : omer Name :	O Pr Navigati Urance aims O Solution 10/02/2021 Marry 10/02/2021 Bobo 10/02/2021 Bobo 10/02/2021
] Custom Pynamic II Loners See Number : : Number : : : Number : : : Number : : : Number : : : Number : : : Number : : : Number : : : Number : : : Number : : : Number : : : : Number : : : : : : : : : : : : : : : : : : :	Image: Cash back scr rer Cash back scr Image: Cash back scr	16:50	Figure 3	Customer Customer Dynamic Insu atomers Unsolved Cla All Claims omer Name : omer Name :	O br Arrow light Urance Image:
Custom ynamic I ymers sei umber : umber : umber : umber : umber : umber :	Image: Cash back scr	reen	Figure 3	Custome - Custome Dynamic Insu atomers Unsolved Cla All Claims omer Name : omer Name :	O or Navigat or Navigat urance urance or Navigat aims O in/02/2021 Marry 10/02/2021 Jelius 10/02/2021 Bobo 10/02/2021 Marry 10/02/2021 Tomas 10/02/2021 Tohmas 10/02/2021
namic II nors ser mber : mber : mber : mber : mber : mber : mber :	Image: Cash back scr	reen	Figure 3	Customer Customer Customer Customer Unsolved Cla All Claims omer Name : omer Name :	O or or urance or or aims O John 10/02/2021 John 10/02/2021 Jobo 10/02/2021 Bobo 10/02/2021 Marius 10/02/2021 Marius 10/02/2021 Tomas 10/02/2021 Harius 10/02/2021

0 Figure 4 - Client (Management of Customer) screen

 \bigtriangledown

Figure 5 - Client (Management of Claims) screen

0 🗆

 \bigtriangledown

UI final product

The Ui did not have to change much except for Client-Side operations.

al ବେ ²⁷ ୪୪ ଓ IQI (୭୫) + 17:38	al ରି 😵 🖏 😻 🛱 💷 +17:38	⊿l ଲ ⁷⁷ Bis to 10 छि। ∳ 17:39
Welcome to Dynamic Insurance	← Get a Quote	
	Deb : 02/04/2004	190@gmail.com Confirm
	000.02/04/2004	
	Hospital : Public Hospitals	Password :
email@gmail.com	2 Cover : Low	First Name : Jonny
password		Last Name : Bravo
Oliest Bar	? Fee :0	
Cilent Reg. Lug II	Plan : Low	Phone Nr : only numbers
Get a Quote	Smoker :	Add address please
		Register
	Get Quote	_
Figure 6, Log in page	Figure 7, Quote page	
		Figure 8, Customer Registration page
⊿∥ रू, ⁰ , ৬ छ छ ।Ω। (छ) + 17:39	리 중, ⁹⁴ 86 전 101 연관 + 17:51	⊿। 😤 290 8/s b/s b /s b /s
		😑 Step Tracker
House Nr : 0	€49.6	
Enter your street name here		
County : <u>co.Dublin</u>	Please enter your card details	
City : Dublin	VISA 4242	
Country : 4 length	Expires End : MM / YY	61,93% Completed
Zip/Post Code : R91 VNX5	Security Code : 123	
Cancel Save	Post Code : R93V8N6	
	Proceed with payment	
		Total Earnad - 0.005
Figure 9, Customer Registration address	Figure 10, Stripe Payment Page	Figure 11, Customer Home page

al रू ⁴⁶ छ छ ।⊡। छ्य)+17:52	al 完 72 w ♂ ℃ 101 930+17:55	al 😤 3 😿 ଔ Ū. 100 + 18:19
LOGOUT		
	Step done last 7 days	
	8620 1069 1842 8703 4607 15222	
健 Home		First Name · Mino
🕒 Profile		
📓 Report		Last Name : Erro
Tlaim		Phone Nr : 77778977
Policy	SUM TU MITER TO	Click to update address
∲° Settings	oday iday nursday ednesday tesday tesday onday unday	Update info
	Steps in the last month	
Current Date	8123	
April 02, 2022		
		\triangleleft \bigcirc \Box
Figure 12, Customer Navigation	Figure 13, Customer Reports	Figure 14, Customer profile page
⊿া হে ৣ ⁶⁴ ৬৫ ত গD। ৫০০০ + 18.20	⊿ ি হ টা	⊿িহি ⁶⁰ ₩ উউটিটিটেট+18:23
I was waking and fall down		Currently under review
		Expiry Date : 28/04/2022
	Currently under review	Hospital: Public Hospitals
	02/04/2022	
Close Submit	Hospital Code : HfIGJ6F8	Cover: Low
	Patient Nr · IG8GGI	Fee: 0
Add Extra info		Plan: Low
┉ ;;	View Extra info	Smoker : 🔽
\times with with the on	All Claims Create	Price : 74.4
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		Update Previous Policies Resolve Update
		\triangleleft \bigcirc \square \otimes
Figure 15, Claims extra info	Figure 16, Customer Claim page	Figure 17, Client resolves selected policy page



Figure 19, The Client manages c customer

Domain model



Figure 21 - Domain Diagram

Class Diagram



Figure 7 - Class Diagram

System Sequence diagrams

The section possesses some of the most important System Sequence (SS) diagrams.



Figure 23, Log in diagram



Figure 22, Registration diagram



Figure 24, Confirm Email Diagram



Figure 25, Pay diagram







Figure 28, Update/View Customer profile SS Diagram







Figure 30, View/Update Policy SS diagram



Figure 31, Get a Quote SS Diagram

Database Models/Schema

The database used by the project is NoSQL therefore these are schema classes that are part of the client's application which reflects the JSON schema that is stored on the cloud database.



Figure 32, Database model diagram

Cloud database schema

The section shows the schema in more detail that is part of the cloud database. The schema is in the type of JSON format which is displayed in the 7 to 12 figures below.

1 -	{
2	"title": "Customer",
3	<pre>"bsonType": "object",</pre>
4 -	"required": [
5	"id"
6],
7 -	"properties": {
8 🔻	"_id": {
9	<pre>"bsonType": "string"</pre>
10	},
11 🔻	"_partition": {
12	<pre>"bsonType": "string"</pre>
13	},
14 -	"Address": {
15	"title": "Address",
16	<pre>"bsonType": "object",</pre>
17	"required": [],
18 🔻	"properties": {
19 🔻	"City": {
20	<pre>"bsonType": "string"</pre>
21	},
22 🔻	"Country": {
23	<pre>"bsonType": "string"</pre>
24	},
25 🔻	"County": {
26	"bsonType": "string"
27	},
28 🔻	"HouseN": {
29	"bsonType": "long"
30	},
31 🔻	"PostCode": {
32	"bsonType": "string"
33	},
34 🔻	"Street": {
35	"bsonType": "string"
36	}
37	}
38	},

"Dob": { 39 -40 "bsonType": "date" 41 }, "Name": { 42 -43 "bsonType": "string" 44 }, "LastName": { 45 🔻 "bsonType": "string" 46 47 }, "PhoneNr": { 48 🔻 "bsonType": "string" 49 50 }, "Email": { 51 -"bsonType": "string" 52 53 }, 54 -"Policy": { "bsonType": "array", 55 "items": { 56 🔻 "bsonType": "objectId" 57 58 } 59 }, 'Reward": { 60 🔻 "bsonType": "array", 61 "items": { 62 -"bsonType": "objectId" 63 64 } 65 }, 'Claim": { 66 🔻 "bsonType": "array",
"items": { 67 68 🔻 "bsonType": "objectId" 69 70 } 71 }, "DelFlag": { 72 🔻 "bsonType": "bool" 73 74 75 🔻 "DataSendSwitch": { "title": "DataSendSwitch", 76 "bsonType": "object",
"required": [77 78 🖛 79 "Switch", "changeDate" 80 81], "properties": { 82 -83 🔻 "Switch": { "bsonType": "bool" 84 },
"changeDate": {
 Type": " 85 86 -"bsonType": "date" 87 88 } 89 } 90 91 } 1 92

Figure 33, Customer object, cloud database schema (part1 & part2)

```
{
   "title": "Claim",
   "' "obje
   "bsonType": "object",
   "required": [
     "_id",
"Accepted",
     "OpenStatus"
   ],
   "properties": {
     "_id": {
    "bsonType": "objectId"
      '_partition": {
    "bsonType": "string"
     },
"StartDate": {
    "bsonType": "date"
      },
      "Accepted": {
        "bsonType": "bool"
      "ExtraInfo": {
    "bsonType": "string"
      "DelFlag": {
        "bsonType": "bool"
     },
"CloseDate": {
    "bsonType": "date"
     },
"HospitalPostCode": {
    ". "string"
        "bsonType": "string"
      },
      "PatientNr": {
    "bsonType": "string"
      },
      "Type": {
    "bsonType": "string"
      },
      "Owner": {
        "bsonType": "string"
     },
     "OpenStatus": {
        "bsonType": "bool"
}
```

Figure 34, Claim object cloud database schema

```
"title": "MovData",
"bsonType": "object",
"required": [
  "_id"
],
'properties": {
  "_id": {
   "bsonType": "objectId"
  },
   _partition": {
    "
"bsonType": "string"
  },
  "DateTimeStamp": {
    "bsonType": "date"
  },
  "DelFlag": {
    "bsonType": "bool"
  },
   'Owner": {
    "bsonType": "string"
  },
   'AccData": {
    "title": "Acc",
    "bsonType": "object",
    "required": [],
    "properties": {
      "X": {
        "bsonType": "float"
      },
"Y":{
        "bsonType": "float"
      },
"Z": {
       "bsonType": "float"
  },
   Type": {
    "bsonType": "string"
```

Figure 35, MovData object, cloud database schema

}

```
"title": "Policy",
  "bsonType": "object",
  "required": [
    "_id"
  ],
  "properties": {
     "_id": {
      "bsonType": "objectId'
     },
     _partition": {
       "bsonType": "string"
     "DelFlag": {
      "bsonType": "bool"
     "Price": {
      "bsonType": "float"
     "PayedPrice": {
    "bsonType": "float"
     } ,
     Cover": {
      "bsonType": "string"
     "HospitalFee": {
      "bsonType": "long"
     "Hospitals": {
    "bsonType": "string"

     "Plan": {
      "bsonType": "string"
     },
     "Smoker": {
      "bsonType": "long"
     },
     'ExpiryDate": {
    "bsonType": "date"
     "UnderReview": {
       "bsonType": "bool"
     "UpdateDate": {
    "bsonType": "date"
     "Owner": {
      "bsonType": "string"
}
```

Figure 36, Policy object cloud database schema

```
{
   "title": "Client",
"bsonType": "object",
    "properties": {
      "_id": {

    "bsonType": "string"
     },
"_partition": {
    "bsonType": "string"
      "Email": {
         "bsonType": "string"
       },
      "FirstName": {
    "bsonType": "string"
      };
"LastName": {
    "'conType":
      "bsonType": "string"
      },
      "CompanyCode": {
   "bsonType": "string"
      },
      "DelFlag": {
    "bsonType": "bool"
      }
    },
    "required": [
      "DelFlag"
    ]
}
```

Figure 37, Client object cloud database schema

```
"bsonType": "object",
   "required": [
    "_id",
"IsFinish"
   ],
   "properties": {
     "_id": {
      "bsonType": "objectId"
     },
     "_partition": {
    "bsonType": "string"
    "bsonType": "float"
    "bsonType": "array",
"items": {
        "bsonType": "objectId"
      }
     },
     "DelFlag": {
      "bsonType": "bool"
     },
     "FinDate": {
      "bsonType": "date"
    },
"StartDate": {
    "bsonType": "date"
    "bsonType": "string"
    },
"IsFinish": {
    "bsonType": "bool"
  }
}
```



Project Plan

Figure 39 shows how the project was planned in time constraints with numbers for the main events. Underneath the chart, it describes in more detail more about each numbered chart event.



Figure 39, Gantt chart plan

- 1. Submit first research document draft.
 - 1.1 Experiment with phone/watch connection.
 - 1.2 Experiment with sending data from watch to phone.
 - 1.3 Identify types of models stored in a database. (Who is part of what)
- 2 Submit first Functional Specification draft.
 - 2.1 Implement a watch app.
- 3 Submit first Design document Draft.
 - 3.1 Find the way to classify steps from raw accelerometer data.
 - 3.2 Display the steps to the user in real-time.
- 4 Store raw accelerometer data.
- 5 Implement Customer API for quote price classification.
- 6 Implement the Quote page.
- 7 Complete customer log-in/Registration app flow.
- 8 Implement the rest of the customer app functionality.
- 9 Implement Client Registration/Log-in app flow.
- 10 Complete the rest of the Client app functionality.
- 11 Implement Stripe into the app.
- 12 Finalize Customers & Clients functionality. (Example adds confirm email, email notifications etc...)
 - 12.1 Showcase website.
 - 12.2 Project demo screencast.
 - 12.3 Project Report.