User Guide
For Mobeye Mobile Application

Provided by Group 4

Version	Date	Changes
1.0	18/01/2021	Initial version

Contents

Introduction

This project is created for the Dutch company Mobeye which specialize in innovative alarm and telemetry technology. The company was founded in 2008 with the idea to help people and organizations to make their life safer and easier by providing them with a meaningful way to secure, control and monitor their property and devices remotely. Mobeye products have received multiple awards and are also used worldwide in various sectors, including police, the agricultural and medical industry, in building management and industry, by both professional users and consumers. Their in-house Research & Development team designs and develops customer-specific products on request as well. The goal of this project is to design and develop a mobile application that would contribute to the Mobeye existing notification system.

Intended use

The purpose of this document is to provide a guided assistance when installing and using the application that Group 4 has created. Detailed information regarding the used technologies and development techniques would be provided as well so that they may assist the authorized personnel at Mobeye to start and later user the application

Features

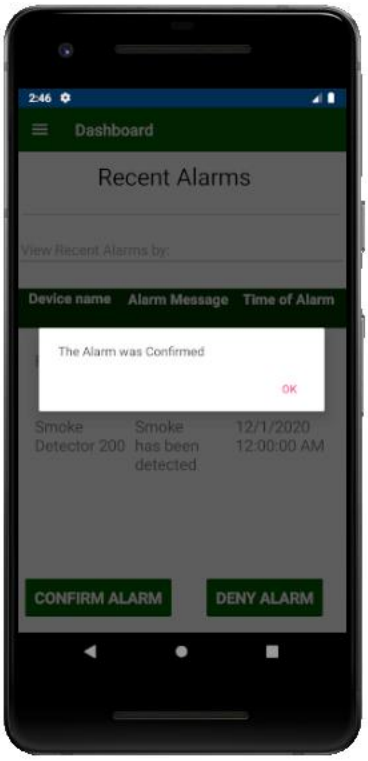
The final product that has been developed by the group has the main functionality that was agreed upon at the beginning. Upon opening the application, the potential end-user inputs a SMS code that was send by Mobeye. It is then immediately checked whether the provided SMS code is correct and the phone unique id is registered in Mobeye's database. As a response to this register request, it is send back a private registration key that is being used in the next immediate call made from the frontend to Mobeye API to check the authorization of this user. When the respective authorization is returned, its corresponding page opens.

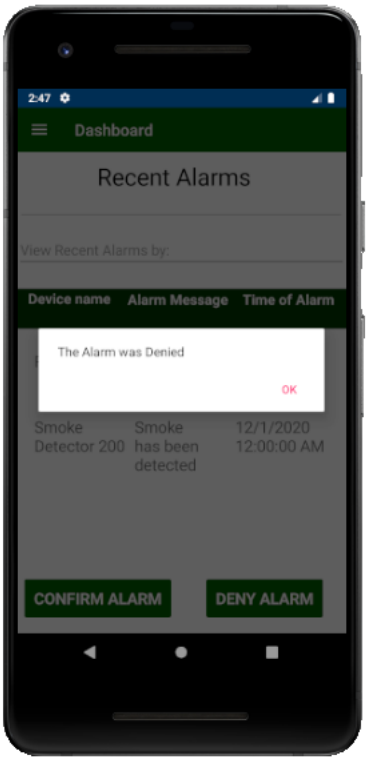
There are 3 types of user authorization: Account User, Contact User and Call-Key User. Each of those users have different functionality and different features in the mobile application.

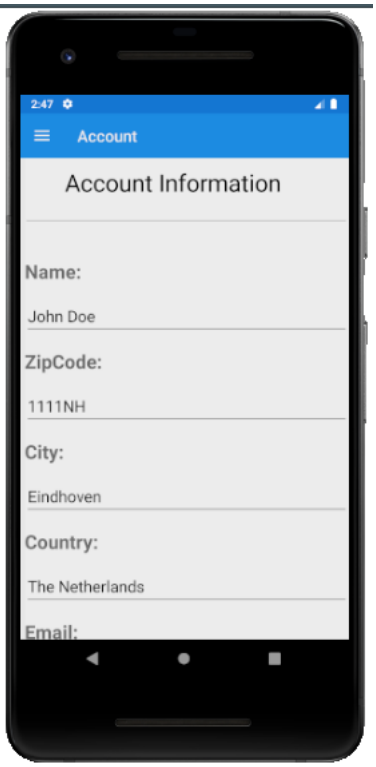
Account User

Firstly, the account user has most functionalities than the other users. He is able to receive all recent alarms, respond to alarms – either confirm or deny them, view his personal information, go to Mobeye's portal and also as an additional feature added by the development team, he would be able to set up his own personal code for accessing the application.


2:38
Dashboard
Recent Alarms
View Recent Alarms by:
Device name
Alarm Message
Time of Alarm
Fire Detector
Fire detector was deactivated
11/30/2020 12:00:00 AM
Smoke Detector 200
Smoke has been detected
12/1/2020 12:00:00 AM
CONFIRM ALARM
DENY ALARM

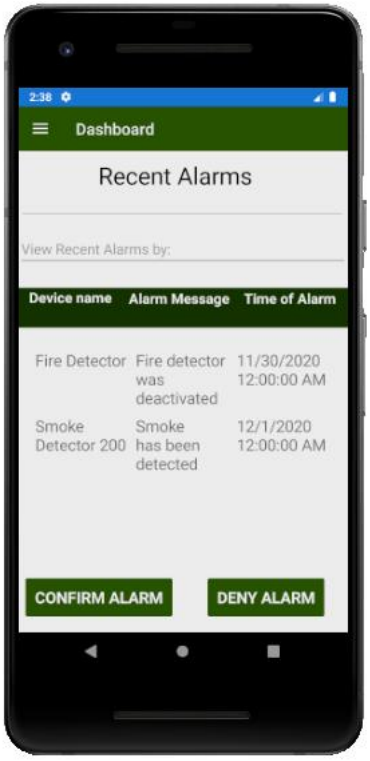

2:46
Dashboard
Recent Alarms
View Recent Alarms by:
Device name
Alarm Message
Time of Alarm
The Alarm was Confirmed
OK
Smoke Detector 200
Smoke has been detected
12/1/2020 12:00:00 AM
CONFIRM ALARM
DENY ALARM

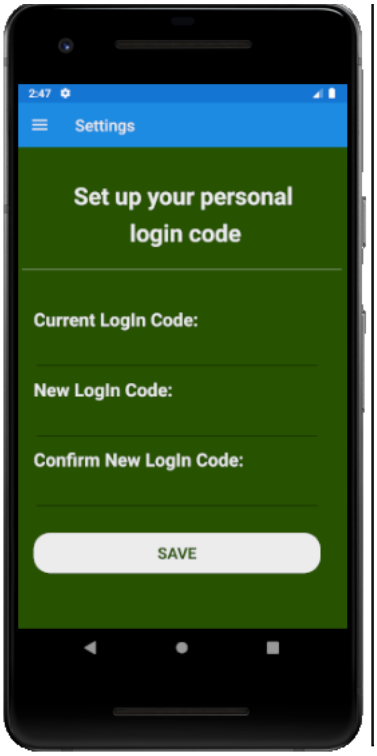

2:47
Dashboard
Recent Alarms
View Recent Alarms by:
Device name
Alarm Message
Time of Alarm
The Alarm was Denied
OK
Smoke Detector 200
Smoke has been detected
12/1/2020 12:00:00 AM
CONFIRM ALARM
DENY ALARM


2:47
Account
Account Information
Name:
John Doe
ZipCode:
1111NH
City:
Eindhoven
Country:
The Netherlands
Email:

Contact User

Secondly, the contact user has almost the same functionality and features as the account user, however he does not have a Mobeye account in their portal, that is why he cannot access Mobeye portal nor view his personal information. However the contact user can still view all recent alarms and respond to them – confirm/deny. Furthermore, both the contact user has the ability to set personal codes, so that he can access the application and also have a second layer of security.

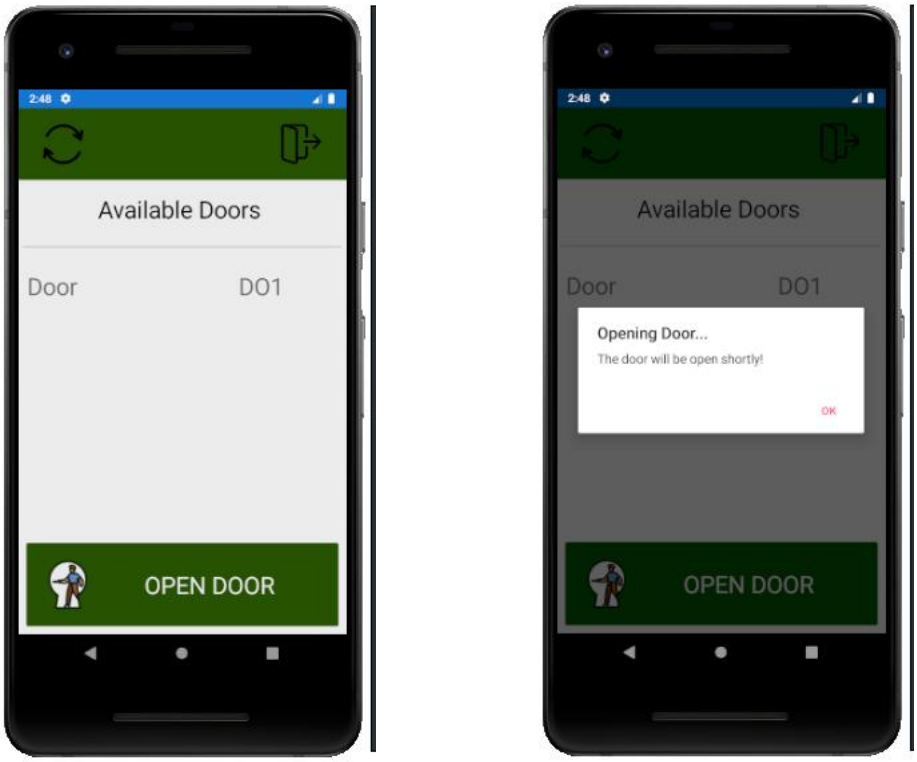




Call-Key User

Lastly, the call-key user has the ability to open doors with his device. When opening the application a page with the available doors is shown and he can open them with just a click of

the button. When the request has been received by Mobeye's API an appropriate message is send back to the user to let him know that everything is working smoothly.

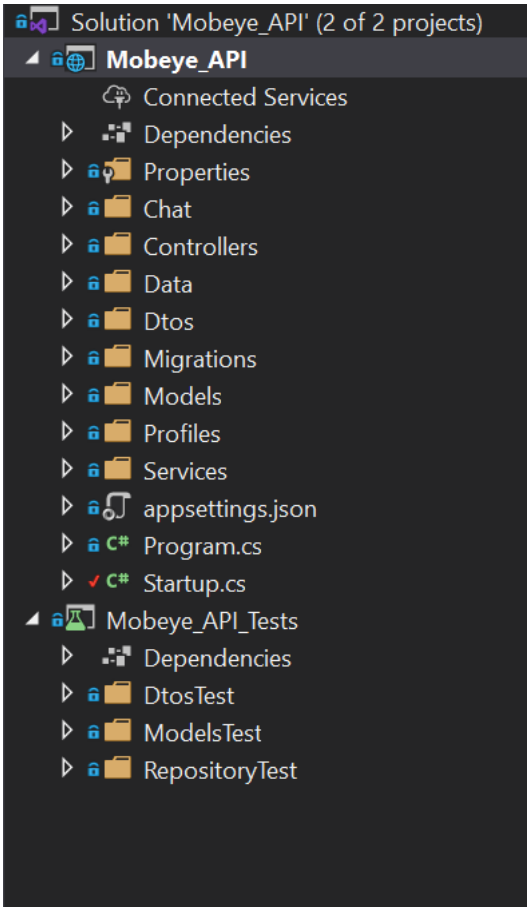


Description of the API

The first part is the API, which would receive calls from Mobeye API. The API is build using ASP.Net Core. Those calls would be later redirected to the frontend, that would display them in an appropriate manner. The calls that would be made from Mobeye API to Group 4's API are regarding the alarms. However, since the group has not been provided with endpoint to authorize and/or test this call, the group simulates a creation of an alarm and sends it to the frontend from their API. In order to integrate the Mobeye API in this process, it would be needed to change on Group 4's API some configurations. That would be, when an alarm is received, instead of retrieving it from the local/remote database, the method responsible for sending the alarms to the frontend would have to configured that it redirects the call that it has received from Mobeye API to the frontend of the application.

The API also has the ability to completely replace Mobeye API if necessary. The API can create devices, alarms and new users with different authorization.

The API is divided into different components and is implemented following the SOLID



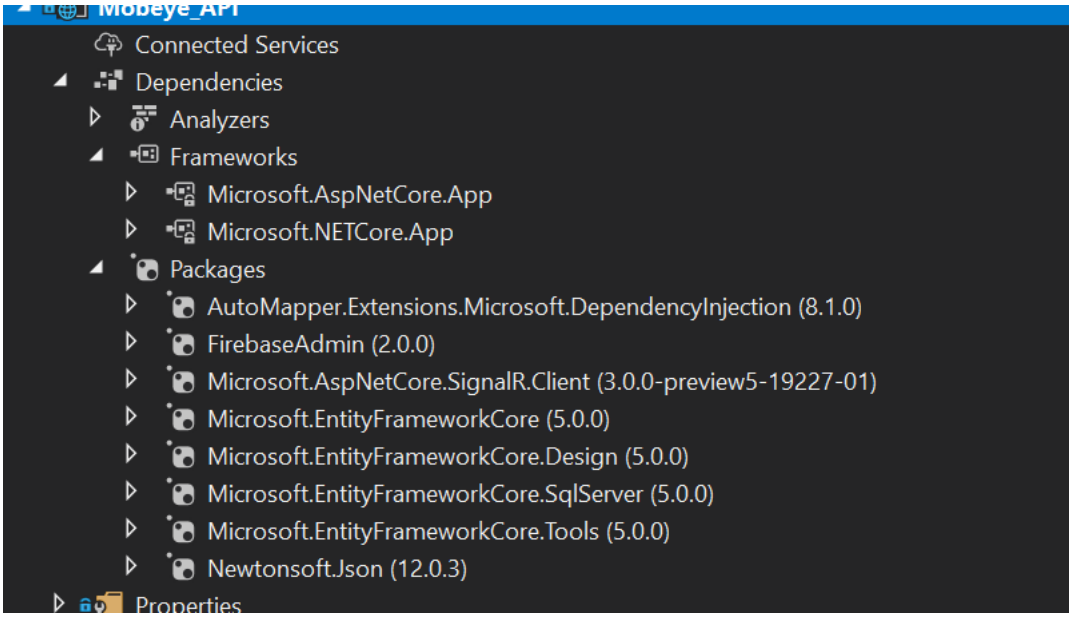
principles so that the individual components are not tightly coupled together.

The Models folder consists of the models that are used to create the tables of the local and remote database the group is using.

The Dtos folder, data transfer object folders consist of objects that are used to encapsulate data and send it from one subsystem of the application to another. They are used to transfer data between API and the UI layer (the frontend Xamarin application).

The Data folder consists of all interfaces and implementation of those interfaces that are concerned with the functionality of the application. The Profiles folder is where the mapping between the models and data transfer objects is happening. Finally, the controllers ' folder consists of the controllers for the API that handle the request-response pipeline of the application with the frontend.

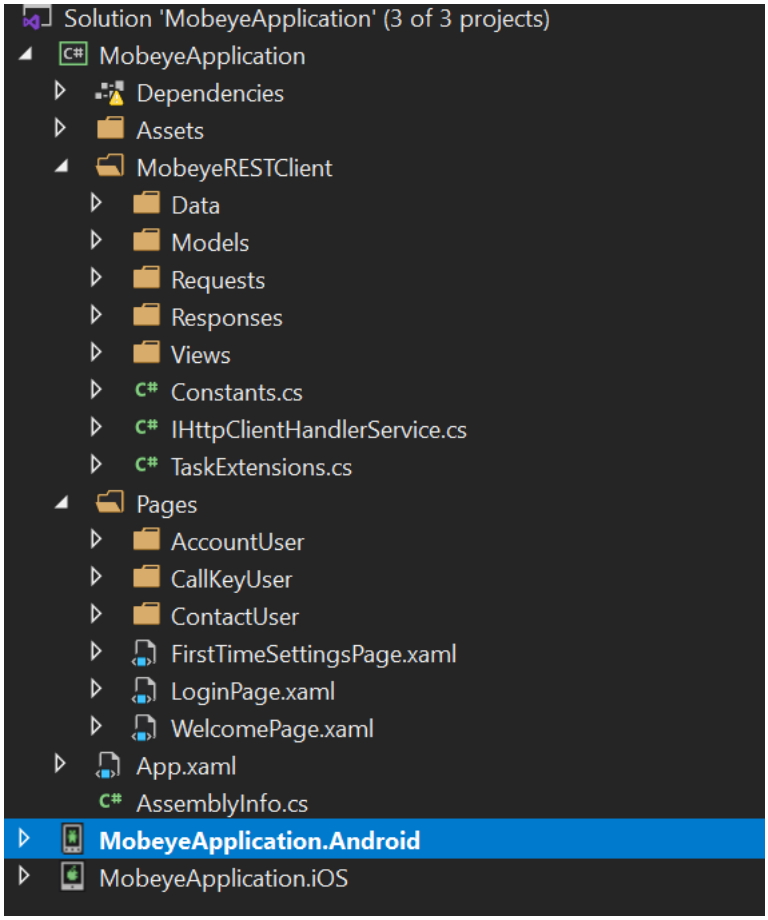
The API has also thorough passing unit and integration testing to guarantee that the application is working properly and would not cause any issues in the future when it is integrated to the Mobeye system.



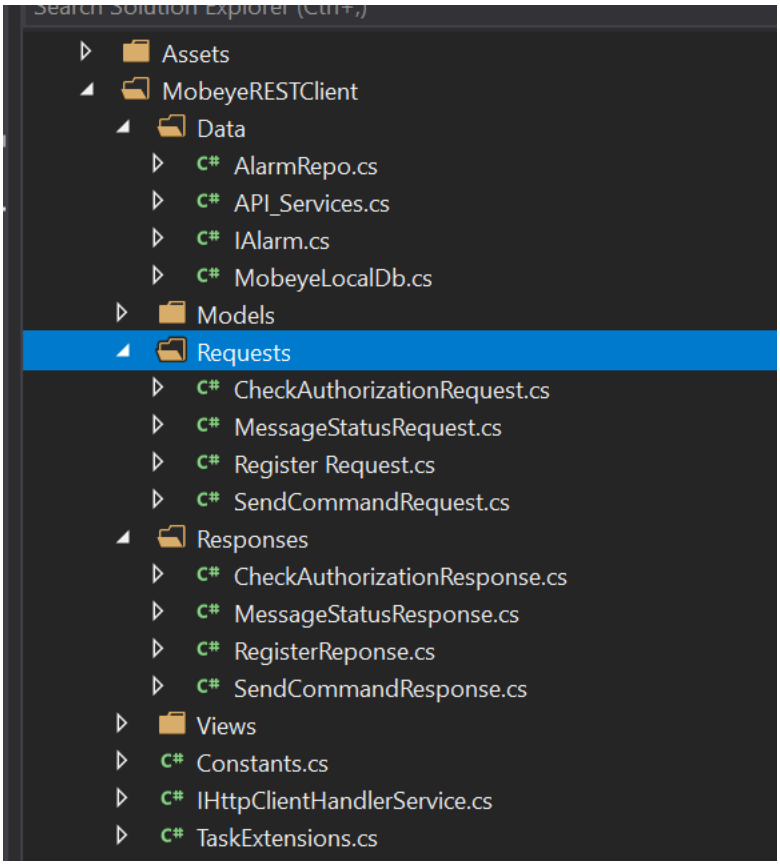
In the picture above all of the frameworks and external dependencies could be seen.

Description of the user interface

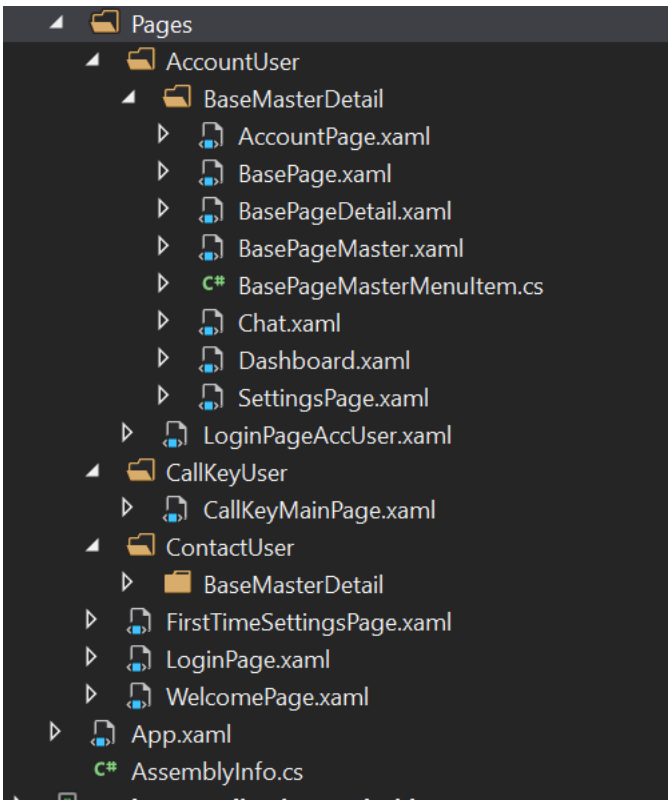
The frontend part of the application is implemented using Xamarin, which is a mobile app platform used for building native applications. The frontend application is created using xml to make the user interface and for the logic behind the application, C# is the main programming language The Xamarin Mobeye application is divided into a logic layer, than handles the request- response pipeline with Mobeye API and with Group 4's API, and a UI layer that is responsible for the user-friendly interface.



The MobeyeRESTClient folder is the place where all the logic of the application is centered at. The Models folder, consists of classes that are used later to bind them to the response that would be received from Mobeye. Requests folder has all the classes with their corresponding POST requests. Responses folder consists of classes with the fields that would be returned from each corresponding request. Data folder consists of 2 important classes – API_Services that handles the communication and the request-response pipeline with Mobeye's API and AlarmRepo that handles the communication and gets the alarms from Group 4's API.



The UI layer of the application is positioned in the Pages folder. There the different views based on the user role. The account user, contact user and call-key user have separate views, where their functionality is implemented and their GUIs are created. There are 3 pages that are shared for all users and that are:



- The Login Page – where users input their SMS Code and are authorized
- First Time Settings Page – where users can input their personal login code
- Welcome Page – when opening the application for the first time and inputting their SMS code, users are greeted by the Welcome Page that gives them the option to either create a personal login code now or leave that for later.

Account User

The pages for the account user are the following:

- Master Detail Pages that are generated by Xamarin and are responsible for the smooth transition between more complicated elements
- Dashboard Page – where all the recent alarms are displayed and where the user can respond to alarms – confirm or deny them
- Setting Page – where the user can view their personal information and if they can to edit it, with a click of a button they would be send to the Mobeye's portal
- Chat Page – where the user would be able to communicate with other users in their contact list who are also clients of Mobeye – however this part is not implemented and it would not be functional

Contact User

The pages of the contact user are the same as the account user with a small difference that the contact user does not have a settings page and is not able to access Mobeye's portal

Call-Key

The page for the call-key user displays the available doors that the user has access to and has a button which when clicked sends a POST request to Mobeye's API that the user wants to open a door and returns a response. When this response it returned, the frontend application displays a message that informs the user that the door will be open shortly.

Installation Instructions

In order to start the application, firstly the backend (API of Group 4) should be up and running. After that the frontend Xamarin application could be started. Since Xamarin is a cross-platform application, there is a possibility to run the application in either Android emulator or IOS environment. Out of the box, Xamarin is able to run Android applications using Android SDK, however in order to run Xamarin IOS application, Visual Studio needs to be connected to a Mac that has to have a valid apple account.

References

https://instrktiv.com/en/user-manual-template/