



H2020 - INDUSTRIAL LEADERSHIP - Leadership in enabling and industrial technologies - Information and Communication Technologies (ICT)

ICT-11-2017 Collective Awareness Platforms for Sustainability and Social Innovation – Innovation Action (IA)



CHILD RESCUE

Collective Awareness Platform for Missing Children Investigation and Rescue

D3.3 - ChildRescue Platform, APIs and Mobile App, Release I

Workpackage:	WP3 – ChildRescue Platform Architecture Definition and Implementation
Authors:	SLG, UBITECH, S5
Status:	Final
Date:	29/06/2019
Version:	1.00
Classification:	Public

Disclaimer:











The ChildRescue project is co-funded by the Horizon 2020 Programme of the European Union. This document reflects only authors' views. The EC is not liable for any use that may be done of the information contained therein

ChildRescue Project Profile

Grant Agreement No.: 780938

Acronym:	ChildRescue
Title:	Collective Awareness Platform for Missing Children Investigation and Rescue
URL:	http://www.childrescue.eu
Start Date:	01/01/2018
Duration:	36 months

Partners

	National Technical University of Athens (NTUA), Decision Support Systems Laboratory, DSSLab <u>Co-ordinator</u>	Greece
	European Federation for Missing and Sexually Exploited Children AISBL - Missing Children Europe (MCE)	Belgium
	The Smile of the Child (SoC)	Greece
	Foundation for Missing and Sexually Exploited Children – (Child Focus)	Belgium
	Hellenic Red Cross (REDCROSS)	Greece
	Frankfurt University of Applied Sciences (FRA-UAS)	Germany
	SINGULARLOGIC ANONYMI ETAIREIA PLIROFORIAKON SYSTIMATON KAI EFARMOGON PLIROFORIKIS (SLG)	Greece
	Ubitech Limited (UBITECH)	Cyprus
	MADE Group (MADE)	Greece
	SUITE5 DATA INTELLIGENCE SOLUTIONS LIMITED (S5)	Cyprus

Document History

Version	Date	Author (Partner)	Remarks
0.1	07/06/2019	Aggeliki Tsakiri (SLG)	ToC
0.2	11/6/2019	Danai Vergeti, George Vafeiadis (UBITECH)	Mobile App
0.3	14/6/2019	Artemis Karlatira, Giannis Varouhos (SLG)	APIs and Backend Integration
0.4	18/6/2019	Dimitris Bikas, Pavlos Katsifarakis, Minas Pertselakis (S5)	Frontend and UI
0.5	24/06/2019	Aggeliki Tsakiri (SLG)	Technical Verification and Evaluation
0.6	26/06/2019	Aggeliki Tsakiri (SLG)	Document Finalisation
0.7	27/06/2019	Danai Vergeti (UBITECH)	Internal Review
0.8	27/06/2019	Minas Pertselakis (S5)	Internal Review
0.9	29/06/2019	Christos Ntanos (NTUA)	Validation of Changes
1.0	29/06/2019	Christos Ntanos (NTUA)	Quality Control

Executive Summary

Deliverable D3.3 – “ChildRescue Platform, APIs and Mobile App Release I” constitutes the supporting documentation of the first iteration of the ChildRescue integrated platform (Release I). It is the output of a combination of WP3 tasks, namely the tasks T3.2 - “Platform Development and Deployment”, T3.3 – “Mobile App Design and Development” and T3.4 – “ChildRescue Platform Technical Verification. D3.3 accompanies the delivered software up to M18 and provides a comprehensive view on the implemented functionalities of the first release of the ChildRescue platform and the mobile app, along with the technical verification and the corresponding evaluation plan.

The presentation of the integrated platform is divided between the core platform and the mobile application. The first platform release generally followed the initial integration plan of D3.1 – “ChildRescue Architecture and Platform Design”, regarding foreseen functionalities and user roles. However, the development of the platform and the mobile app led to minor modifications and adaptations of the original architecture which are highlighted in this document. A brief section for each of the implemented components follows, including a short description of each entity, its integration in the platform and some indicative screenshots. The required APIs for information retrieval and exchange are referenced inside the integration section of each component. However, for a complete reference of all implemented ChildRescue APIs for Release I, a separate section is dedicated to the API documentation. The last part of the integrated ChildRescue platform presentation is the first release of the mobile application, with the relevant description and screenshots.

The second thematic unit of this document is the outline of the ChildRescue technical verification and evaluation plan, which will set the processes and criteria for verification of functional and non-functional requirements’ fulfilment and delivery of high-quality code. The technical verification and evaluation plan have been put into effect right from the start of implementation activities. This first release has been verified to comply with high-quality standards, and some of the results from completed tests are presented indicatively in this document.

The delivery of Release I signifies the completion of the second platform development iteration and the start of the third and final one. The final iteration includes further development and finetuning of already delivered components, implementation of those that were scheduled for the second release, testing and technical evaluation. WP4 activities, which will validate the platform in terms of added end-user value, will take place in parallel to the ongoing platform development and provide valuable feedback. Release II will be delivered in the context of D3.4 – “ChildRescue Platform, APIs and Mobile App Release II” in [M27], while a full public release is due for launching in M29.

Table of Contents

1	Introduction	9
2	ChildRescue Integrated Platform Release I.....	10
2.1	General Status and Architecture Updates	10
2.2	Core Platform and Components.....	12
2.2.1	Case Manager Component.....	13
2.2.1.1	<i>Description</i>	<i>13</i>
2.2.1.2	<i>Integration</i>	<i>13</i>
2.2.1.3	<i>Screenshots.....</i>	<i>14</i>
2.2.2	Control Room	17
2.2.2.1	<i>Description</i>	<i>17</i>
2.2.2.2	<i>Integration</i>	<i>18</i>
2.2.2.3	<i>Screenshots.....</i>	<i>18</i>
2.2.3	Privacy, Anonymisation, Synchronisation and Security Engine	18
2.2.3.1	<i>Description</i>	<i>18</i>
2.2.3.2	<i>Integration (may also include APIs).....</i>	<i>18</i>
2.2.3.3	<i>Screenshots.....</i>	<i>19</i>
2.2.4	Notification Engine	19
2.2.4.1	<i>Description</i>	<i>19</i>
2.2.4.2	<i>Integration</i>	<i>19</i>
2.2.4.3	<i>Screenshots.....</i>	<i>19</i>
2.3	APIs.....	19
2.3.1	Mobile User Endpoints.....	20
2.3.1.1	<i>Mobile Alert Endpoints.....</i>	<i>20</i>
2.3.1.2	<i>Mobile Case Endpoints</i>	<i>20</i>
2.3.1.3	<i>Mobile Feedback Endpoints.....</i>	<i>20</i>
2.3.1.4	<i>Mobile User Endpoints.....</i>	<i>21</i>
2.3.2	Web User Endpoints.....	22
2.3.2.1	<i>Web Alert Endpoints.....</i>	<i>22</i>
2.3.2.2	<i>Web Case Endpoints</i>	<i>23</i>
2.3.2.3	<i>Web Facility Endpoints</i>	<i>24</i>
2.3.2.4	<i>Web Feedback Endpoints.....</i>	<i>25</i>
2.3.2.5	<i>Web Organisation Endpoints.....</i>	<i>25</i>
2.3.2.6	<i>Web User Endpoints.....</i>	<i>26</i>
2.4	Mobile Application	26
2.4.1	Description	26

2.4.2	Integration	28
2.4.3	Screenshots.....	30
2.4.3.1	<i>Welcome</i>	<i>30</i>
2.4.3.2	<i>Login-Register</i>	<i>31</i>
2.4.3.3	<i>Dashboard.....</i>	<i>31</i>
2.4.3.4	<i>Provide Information for specific alert.....</i>	<i>32</i>
2.4.3.5	<i>Provide Information</i>	<i>33</i>
3	ChildRescue Integrated Platform Technical Verification and Evaluation	34
3.1	Technical Verification	34
3.1.1	Integration Testing	34
3.1.1.1	<i>Approaches and Techniques</i>	<i>34</i>
3.1.2	Integration Points	36
3.1.2.1	<i>User Registration</i>	<i>36</i>
3.1.2.2	<i>Receive Alert for specific case.....</i>	<i>38</i>
3.1.2.3	<i>Send feedback for specific case</i>	<i>42</i>
3.2	Technical Evaluation	45
3.2.1	Product Quality Model Categories.....	45
3.2.2	Technical KPIs of the ChildRescue Platform	47
4	Next Steps.....	51
	Annex I: Implemented User Stories Release I	52

List of Figures

Figure 2-1: Updated High-level Architecture Diagram	11
Figure 2-2: Case/Facility Management – Children Cases	14
Figure 2-3: Case/Facility Management - Case Information	14
Figure 2-4: Case/Facility Management - Edit Case	15
Figure 2-5: Case/Facility Management - Feedback Overview	15
Figure 2-6: Case/Facility Management - Update Feedback	16
Figure 2-7: Case/Facility Management - Manual Feedback Evaluation	16
Figure 2-8: Case/Facility Management - Alert Overview	17
Figure 2-9: Case/Facility Management - Update Alert	17
Figure 2-10: Control Room – Case/Facility Dashboard	18
Figure 2-11: Platform Login	19
Figure 2-12: ChildRescue Mobile App Technical Components	28
Figure 2-13: Mobile App Requests New Content.....	29
Figure 2-14: Mobile App Gets New Case Alert	29
Figure 2-15: Mobile app Receives New Notification.....	30
Figure 2-16: Welcome Description.....	31
Figure 2-17: Welcome Title.....	31
Figure 2-18: Login	31
Figure 2-19: Register Example	31
Figure 2-20: Dashboard - Registered User	32
Figure 2-21: Profile - Registered User	32
Figure 2-22: Provide Information-Complete Form (1/2)	32
Figure 2-23: Provide Information - Complete Form (2/2)	32
Figure 2-24: Provide Information Step 1	33
Figure 2-25: Provide Information Step 2	33
Figure 2-26: Provide Information	33
Figure 3-1: User Registration Integration Point	36
Figure 3-2 : Receive Alert Integration Point	39
Figure 3-3: Send Feedback Integration Point	42

List of Tables

Table 2-1: Updated Components, Sub-Components and Functionalities	11
Table 2-2: ChildRescue Platform technologies and frameworks	12
Table 2-2: ChildRescue Mobile App Versions	27
Table 2-3: ChildRescue Mobile App technologies and frameworks	28
Table 3-1: User Registration Integration Test	36
Table 3-2: Receive Alert Integration Test	39
Table 3-3: Send Feedback Integration Test	42
Table 3-4: Product Quality Model Categories.....	45
Table 3-5: ChildRescue Technical Evaluation KPIs	47

1 Introduction

D3.3 is released in the context of WP3 – “ChildRescue Platform Architecture Definition and Implementation” and contains the first complete integrated platform release, which is the result of the development and technical evaluation activities of tasks T3.2, T3.3 and T3.4 until [M18]. The deliverable is of type “Demonstrator”, meaning that it consists of the actual developed software and the relevant technical system documentation.

The present document consolidates elements from three types of technical documentation: 1. software architecture - the complete presentation of the core platform along with each implemented component, and the mobile application, 2. API documentation - the API calls and the corresponding components’ endpoints and 3. quality assurance documentation - the first version of the technical platform verification and evaluation plan and the results of tests conducted thus far. Lastly, for the sake of completeness, implemented user stories are listed in Annex I: Implemented User Stories Release I.

The main body of the document is structured as follows: the ChildRescue integrated platform is presented in Section 2, and the technical verification and evaluation framework is outlined in Section 3, along with results from performed tests. The deliverable gets its input from previous WP3 deliverables: D3.1, where the platform architecture, functionalities and relationships were designed and detailed based on elicited user requirements and developed user stories, and D3.2 which contained the functional mock-ups for the web and mobile application user interfaces.

The current software release, which is currently hosted in three GitHub repositories, will be further developed in order to deliver all planned functionalities and address all user stories as refined in D3.1. User feedback and evaluation of the pilot phases will serve as a base for updating and improving this release, while the results of the validation and verification processes will lead to delivering high quality software. The next official release is planned for [M27], in the context of deliverable D3.4 – “ChildRescue Platform, APIs and Mobile App, Release II”.

2 ChildRescue Integrated Platform Release I

2.1 General Status and Architecture Updates

The development of functionalities, the corresponding ChildRescue components and the mobile application, as well as their incorporation in the integrated platform, generally followed the design and architecture outlined in D3.1 – “ChildRescue Architecture and Platform Design”. For the first software release, it prescribed the implementation of core platform components enabling the execution of the following actions: registration of organisation and members, registration of independent users, creation and management of case and child profiles, reception of evidence from mobile app users, notification generation and diffusion.

The relevant code is stored into three private (for the time being) repositories in GitHub for better control and error handling:

1. Backend & APIs: <https://github.com/singularlogic/child-rescue>
2. Frontend & UI: <https://github.com/singularlogic/child-rescue-ui>
3. Mobile App: <https://github.com/ubitech/childrescue-mobile>

A demo platform is setup on a private server with access rights available on request for security reasons. The mobile app can be currently deployed as an .apk file, but it will soon be available as a private app through Google store. The ChildRescue consortium has access to both for evaluation purposes. Of course, once the platform and app reach a development status where they can support real users, they will go public. This is estimated to occur on [M29].

Since the beginning of the development, a few technical factors, as well as the feedback and comments from the pilot partners resulted in a number of modifications on the ChildRescue high-level architecture (as presented in D3.1) with its current status being depicted in Figure 2-1.

In more detail, on component level there is the renaming of the “Communication Engine” to “Notification Engine” since this component will handle only the push notifications from the server towards the mobile app or the web UI. Regarding the sub-components of the “Case Manager” component, they have been restructured and modified to accurately depict implemented functionalities. The “Multilayer Profile Creation” and “Profile Update” sub-components have been unified to one. “Tag Management” sub-component has also been incorporated into the “Profile Creation and Update” sub-component in the new version, as its functionality and complexity don’t justify the existence of a separate sub-component. Two new sub-components, whose functionalities were not represented by any other entity in the diagram, were introduced in the “Case Manager”, namely the “Feedback Management” and “Alert Management” sub-components. The “Feedback Management” sub-component is responsible for allowing web users create, update and evaluate feedback, and forward it to the “Evaluation Engine”, “Profiling and Prediction Engine” and “Blockchain” components. The “Alert Management” sub-component is responsible for the content, location/radius, and duration customisation of alerts by the web user. Once a new alert is ready and activated, a signal is sent to the “Notification Engine” for the diffusion of the appropriate notifications to the mobile application users. The “Alert” is a new term, introduced in ChildRescue during the platform implementation, as there was a need to distinct these entities from notifications. Notifications now refer only to mobile push notifications and are sent only during the alert creation and the case closure. At the same time, some functionalities have been moved

from the, previously called, "Communication Engine" to the "Alert Management" sub-component, so that the former now has mainly the role of a message transmitter rather than a configurator.

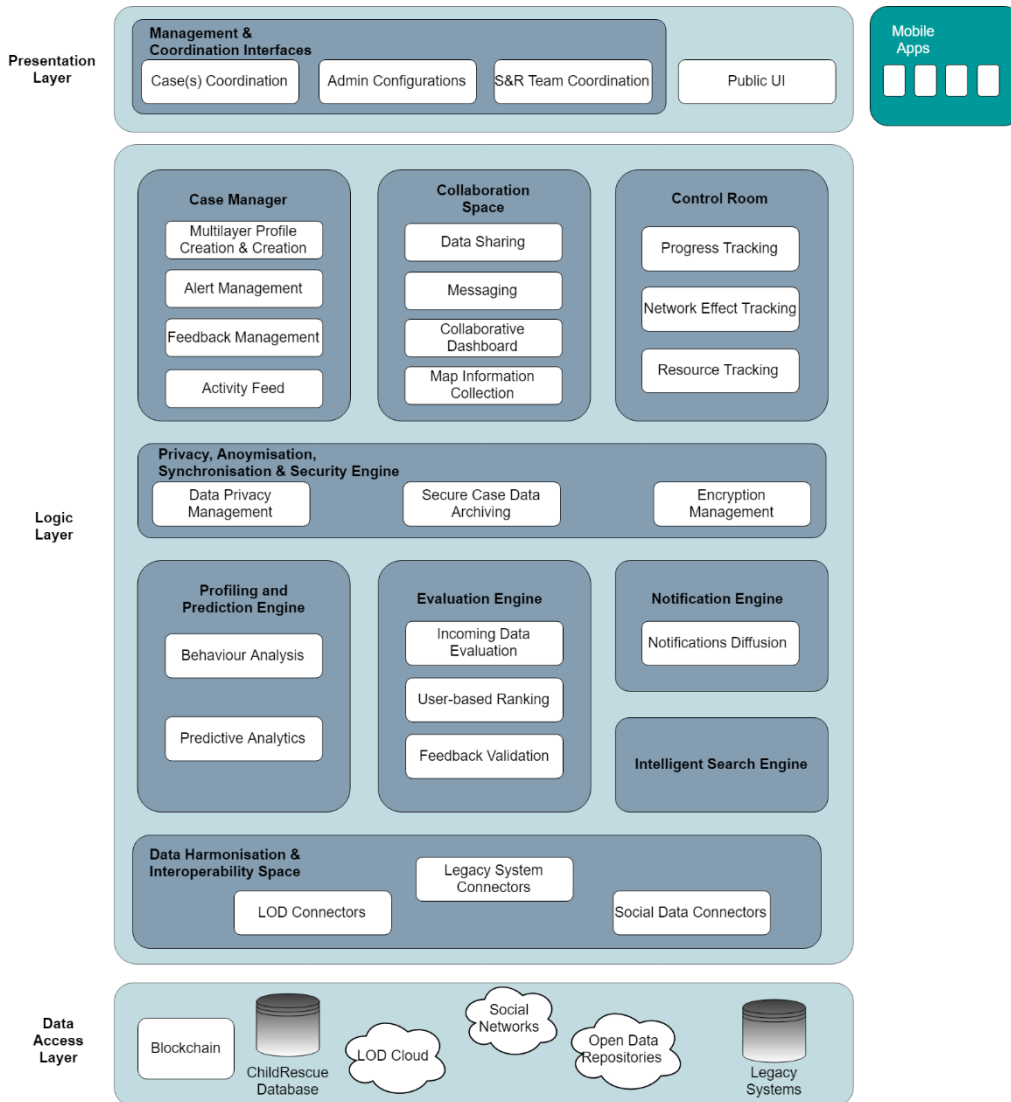


Figure 2-1: Updated High-level Architecture Diagram

The updated components and their functionalities are presented in the following Table 2-1. The original versions of the updated components and architecture, along with the complete descriptions of functionalities and components that are not presented here, can be found in D3.1 – "ChildRescue_D3.1 ChildRescue Architecture and Platform Design_v1.0".

Table 2-1: Updated Components, Sub-Components and Functionalities

ChildRescue Component	Sub-Components	Functionalities	In Release
Case Manager	Multilayer Profile Creation & Update	Multilayer Case Profile Generation	I
		Ongoing Data Acquisition	I
		Profile Enrichment	I

		Tag Retrieval from Collaboration Space & Case Tagging	II
	Alert Management	Alert Configuration, Creation & Update	I
	Feedback Management	Feedback Creation & Update. Manual Evaluation	I
	Activity Feed	Actions Logging	II
Notification Engine	Notifications Diffusion	Target and send messages (push notifications)	I

2.2 Core Platform and Components

The progress made until [M18] regarding the core platform development and its components is in line with the integration plan. The major goal was to develop the appropriate software infrastructure (Backend – Rest API) and setup the data storage (ChildRescue Database) to handle the background functionality and API calls for most of the given actors and actions. From the user interface point of view (Frontend-UI), the focus was set on the Case Management, so that most of the functionalities required to create, update and disseminate a case and relevant notifications (Notification Engine) to mobile end users are available in Release I.

The currently utilised technologies and frameworks are summarised in Table 2-1Table 2-2.

Table 2-2: ChildRescue Platform technologies and frameworks

Component	Implemented Framework
Backend	Python 3.6, Django framework ¹ , version 2.x
Frontend-UI	Vue.js ² , version 2.x
Database	PostgreSQL ³ , version 10.0
Notification Engine	Firebase Cloud Messaging ⁴ (server app)
REST API	Django Rest framework ⁵

¹ <https://www.djangoproject.com/>

² <https://vuejs.org/>

³ <https://www.postgresql.org/>

⁴ <https://firebase.google.com/docs/cloud-messaging>

⁵ <https://www.django-rest-framework.org/>

In the following sections, the implemented ChildRescue components are presented individually: firstly, there is a brief description of the component's functionality, then the integration process and API calls are listed and lastly some indicative screenshots are presented.

2.2.1 Case Manager Component

2.2.1.1 Description

The "Case Manager" component can be used by web users to create and manage multi-layer case profiles, alerts and feedback. Its interface is considered to be the main tool for the Case Manager, the Facility Manager and the Coordinator.

The multi-layer profile information is divided in groups of personal, demographic, physical, medical, psychological, social media data and other basic case information; this information together with outputs from other components in Release II (such as the "Prediction and Profiling Engine") will compose the complete profile of the missing child or the unaccompanied migrant minor. This profile will be updated throughout the case's lifetime.

Feedback can be created and edited through this component. The manager can also proceed to a manual assessment of feedback coming from citizens and manually mark it as relevant, irrelevant or credible.

The "Case Manager" component is also employed when the manager wants to create an alert for the case. The manager can customize the alert's content (e.g. direct data retrieval from the amber alert system, manually add free-text, attach files), duration, location/radius. These alerts can be updated after creation and even be deactivated (where they cannot be edited any more).

2.2.1.2 Integration

The "Case Manager" component was integrated in the ChildRescue Core Platform by expanding the platform's user interface and connecting the engine to the backend. The module's frontend has been integrated using JavaScript and it is based on the built-in modules of Vue.js framework to communicate with a RESTful API.

Concerning the component's API calls there are four endpoints that make API calls: cases, facilities, alerts and feedbacks (evidences). Moreover, there are five basic API calls that every endpoint can perform; list, create, read, update and delete that are self-explanatory. However, it should be mentioned that the user will only be able to perform the calls list, update and read for the accounts of an organisation that he has access to. Additionally, a manager of the ChildRescue platform cannot update or delete a facility of a different organisation. The other endpoints have to do with the permission a user has and as such can be read, listed, created, updated and deleted.

2.2.1.3 Screenshots

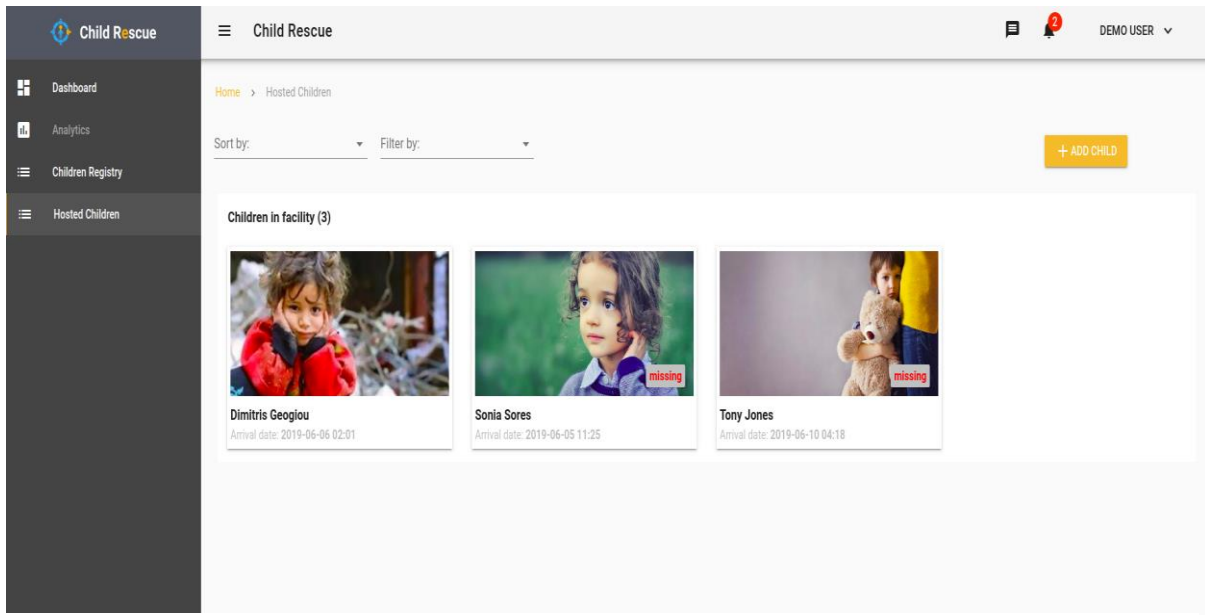


Figure 2-2: Case/Facility Management – Children Cases

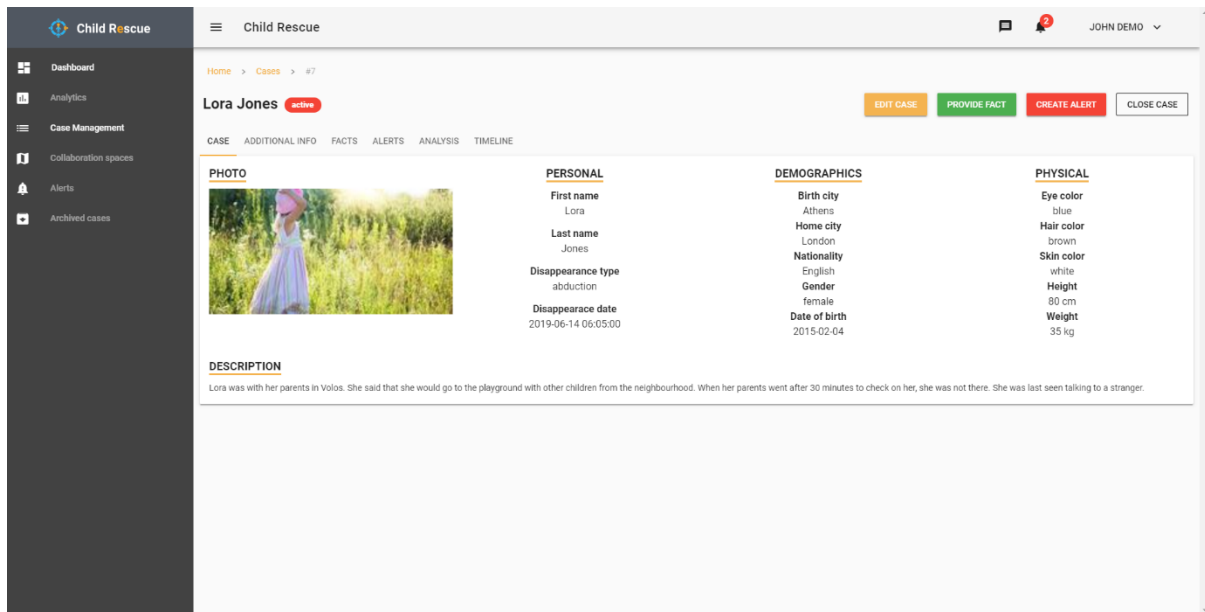


Figure 2-3: Case/Facility Management - Case Information

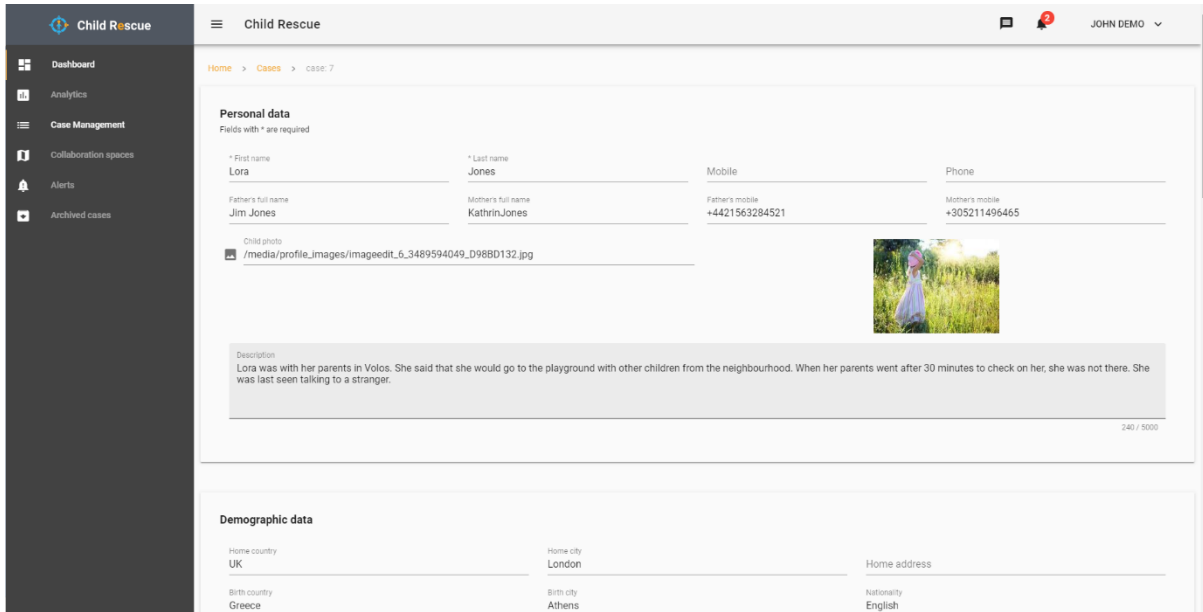


Figure 2-4: Case/Facility Management - Edit Case

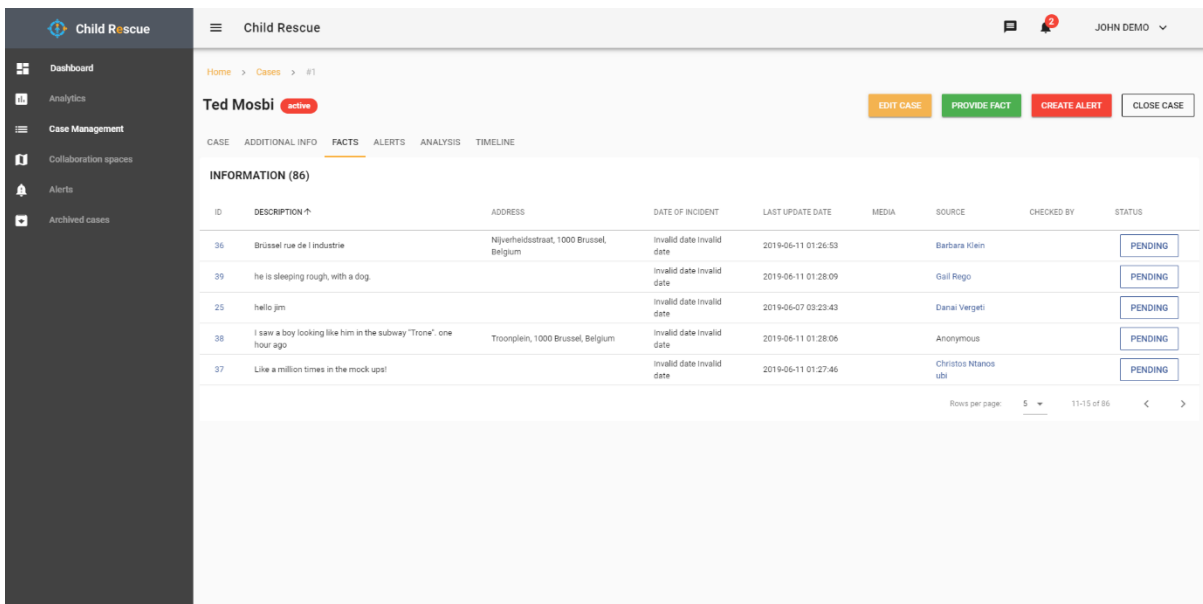


Figure 2-5: Case/Facility Management - Feedback Overview

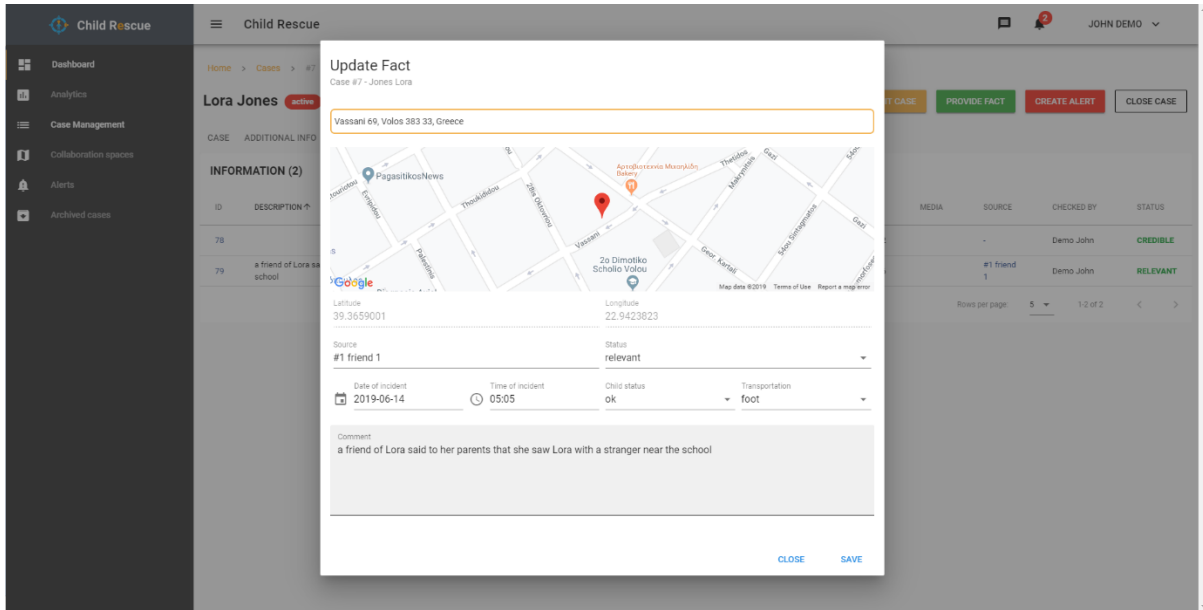


Figure 2-6: Case/Facility Management - Update Feedback

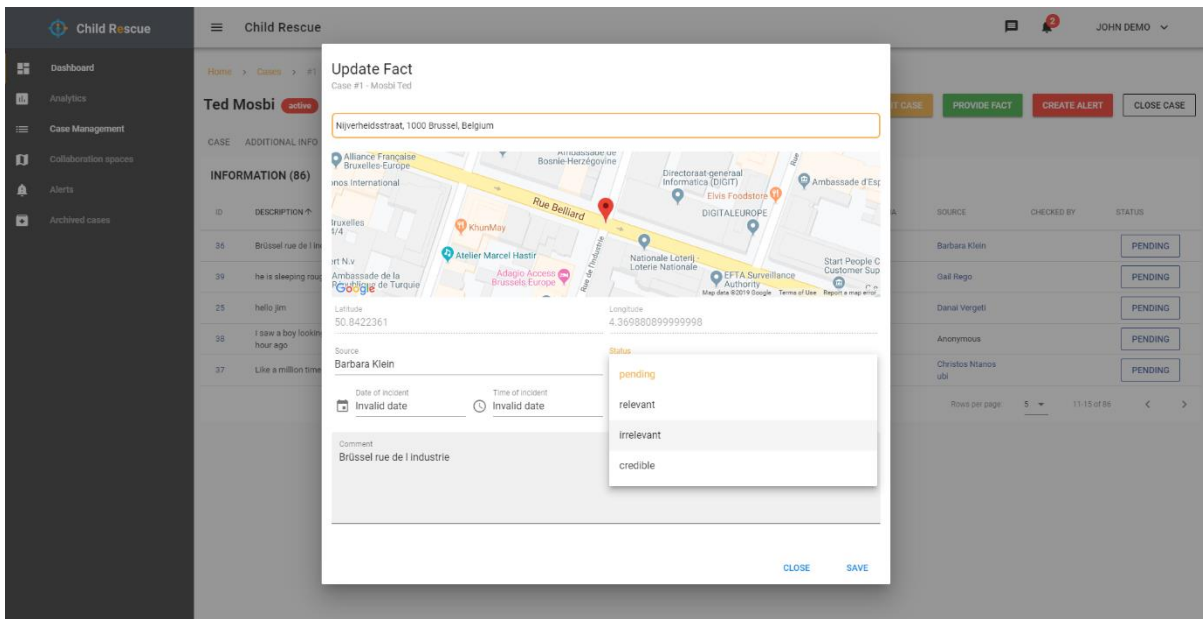


Figure 2-7: Case/Facility Management - Manual Feedback Evaluation

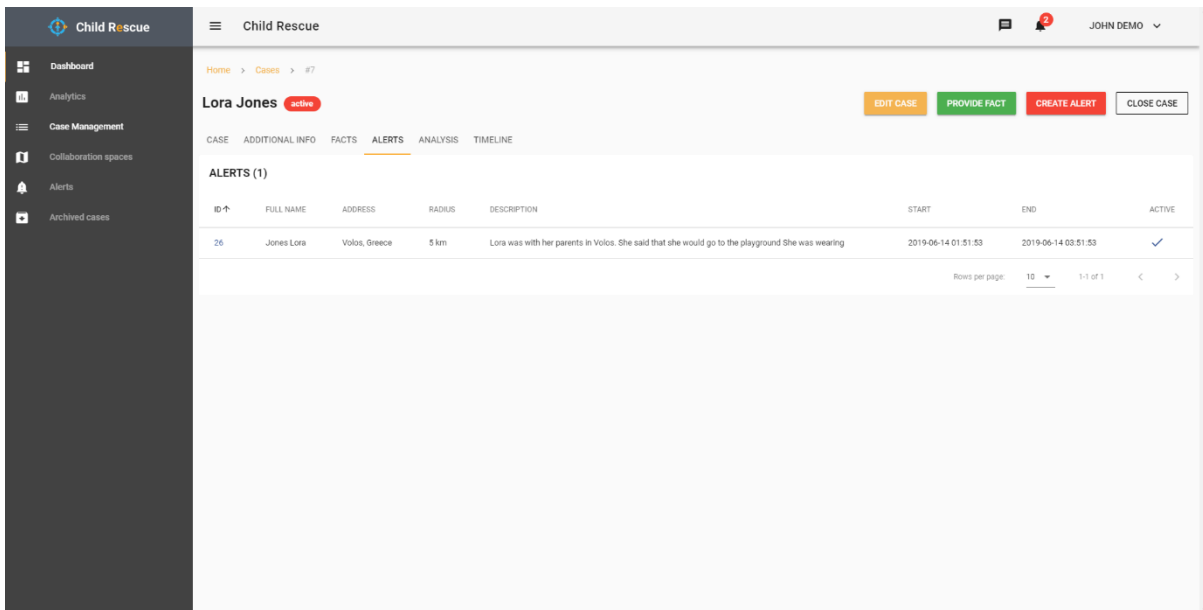


Figure 2-8: Case/Facility Management - Alert Overview

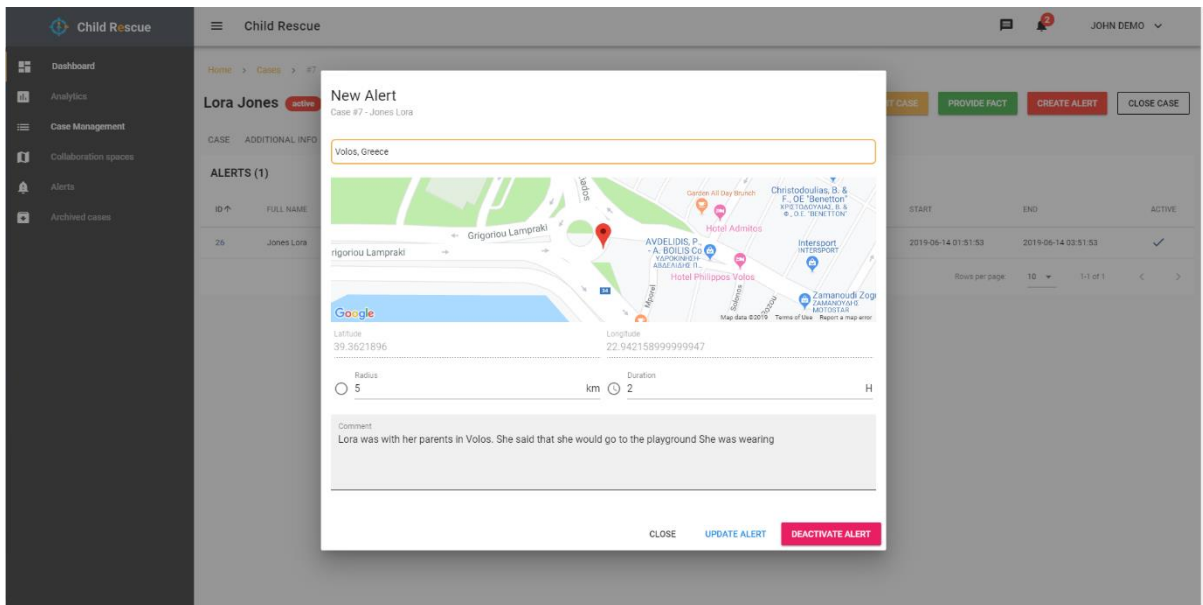


Figure 2-9: Case/Facility Management - Update Alert

2.2.2 Control Room

2.2.2.1 Description

The “Control Room” Component is basically a tool that offers an overview to the manager about what is currently going on and what is urgent to be addressed. As requested by the pilot partners, all Case Managers should be able to view all cases, so in terms of case monitoring, they are not much different than the Organisation Coordinators.

The overview shows some basic statistics, the most informative active cases (based on the number of new feedback) and the recent activity feed, for a quick view of the latest updates regarding all cases.

2.2.2.2 Integration

The “Control Room” Component was integrated in the ChildRescue Platform UI using the Vue.js framework. It retrieves aggregated information about the current statistics, the most active cases in terms of recent feedback, and the most recent activity in the form of a feed.

2.2.2.3 Screenshots

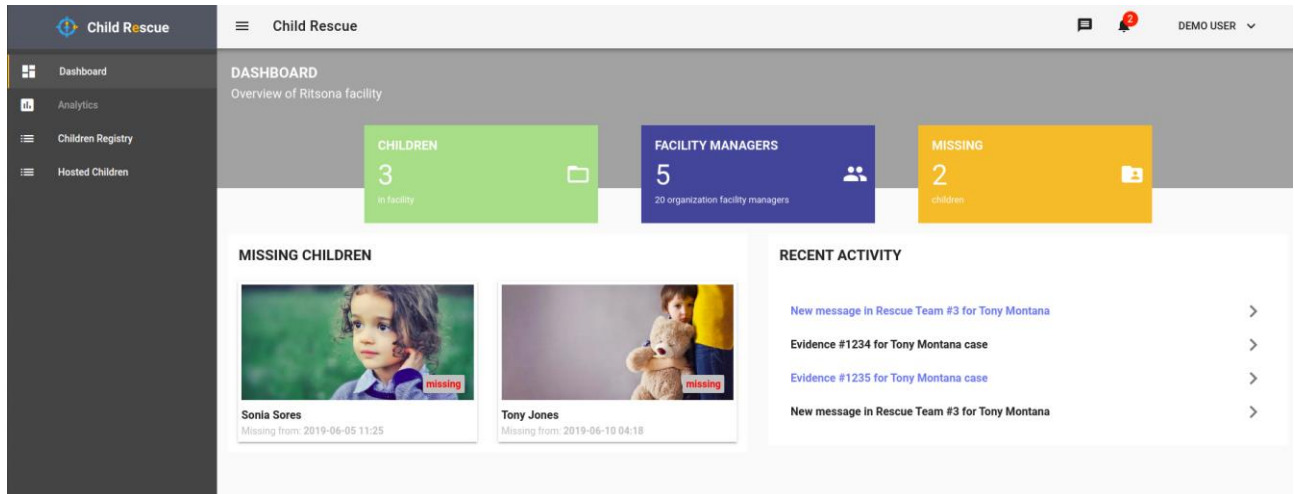


Figure 2-10: Control Room – Case/Facility Dashboard

2.2.3 Privacy, Anonymisation, Synchronisation and Security Engine

2.2.3.1 Description

The “Privacy, Anonymisation, Synchronisation & Security Engine” is a tool used mainly for security purposes to ensure that no sensitive data are released nor stored unencrypted. In particular, for the current release, we use one-way user password credentials encryption to succeed this.

2.2.3.2 Integration (may also include APIs)

The one-way user password credentials encryption is based on the idea to store the hash values of passwords instead of the actual passwords provided by the users. For this purpose, the way that is used to validate the user’s credentials is to apply the hash function to the password submitted by the user and compare it with the stored value in the database. If the hashes match the user is authenticated.

- `login`: Used to *post* user’s email and password to the platform to login and obtain the token to access all other requests and identify the user.
- `register`: Used to *create* new user by providing the name, email, role password and organisation
- `logout`: Used to revoke the token and block the access to the requests
- `me`: Used to *get* information (name, email, organisation, avatar) of the logged user

2.2.3.3 *Screenshots*

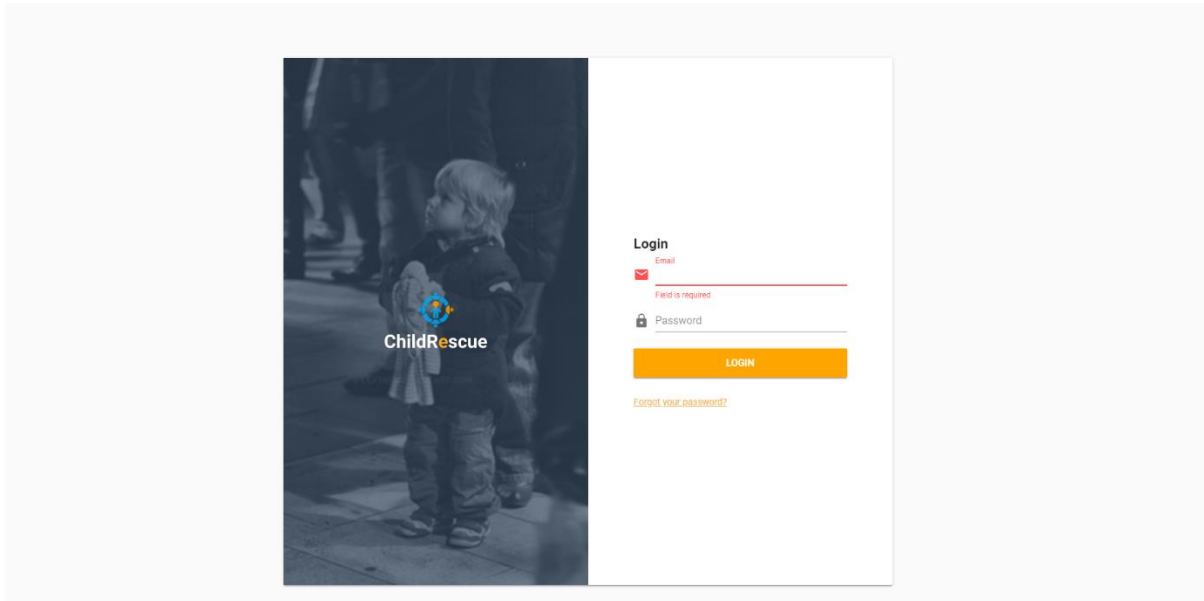


Figure 2-11: Platform Login

2.2.4 **Notification Engine**

2.2.4.1 *Description*

This component materializes the messaging system of the platform and is responsible for the notifications' diffusion to mobile users. It is based on Google's Firebase SDK for server-side functionality. When a new alert is created or when a case is closed, the engine sends the appropriate message to all registered mobile users. The client side of the Firebase checks, then, the mobile's location and selects whether the notification will be shown or not.

2.2.4.2 *Integration*

The "Notification Engine" component is integrated to the Django backend as a unified platform library for sending push notifications to mobile devices and browsers. The library comes with Django REST Framework support.

2.2.4.3 *Screenshots*

Since this is a background service, there are no screenshots available.

2.3 **APIs**

The API Documentation presents the endpoints that are used in order to retrieve information for web and mobile users. This section is divided into two parts, the mobile and the web endpoints, based on the application that sends the request.

2.3.1 Mobile User Endpoints

2.3.1.1 Mobile Alert Endpoints

GET	/mobile/api/v1/alerts/
GET	/mobile/api/v1/alerts/{format}
GET	/mobile/api/v1/alerts/{id}/
GET	/mobile/api/v1/alerts/{id}{format}

A mobile user has access to retrieve alerts based on chronological and geographical criteria.

2.3.1.2 Mobile Case Endpoints

GET	/mobile/api/v1/cases/
GET	/mobile/api/v1/cases/{format}
GET	/mobile/api/v1/cases/{id}/
POST	/mobile/api/v1/cases/{id}/follow_case/
POST	/mobile/api/v1/cases/{id}/follow_case{format}
POST	/mobile/api/v1/cases/{id}/unfollow_case/
POST	/mobile/api/v1/cases/{id}/unfollow_case{format}
GET	/mobile/api/v1/cases/{id}{format}

A mobile user can retrieve cases and choose to follow or unfollow a case he is interested in knowing more in the future.

2.3.1.3 Mobile Feedback Endpoints

GET	/mobile/api/v1/feedbacks/
POST	/mobile/api/v1/feedbacks/
GET	/mobile/api/v1/feedbacks/{format}
POST	/mobile/api/v1/feedbacks/{format}
GET	/mobile/api/v1/feedbacks/{id}/
GET	/mobile/api/v1/feedbacks/{id}{format}

Using the mobile feedback endpoints, the user has the ability to send a feedback as well as read a feedback he has sent in the past.

2.3.1.4 *Mobile User Endpoints*

POST	/mobile/api/v1/users/forgot-password/
POST	/mobile/api/v1/users/forgot-password{format}
POST	/mobile/api/v1/users/login/
POST	/mobile/api/v1/users/login{format}
POST	/mobile/api/v1/users/logout/
POST	/mobile/api/v1/users/logout{format}
GET	/mobile/api/v1/users/me/
PUT	/mobile/api/v1/users/me/
PATCH	/mobile/api/v1/users/me/
DELETE	/mobile/api/v1/users/me/
GET	/mobile/api/v1/users/me{format}
PUT	/mobile/api/v1/users/me{format}
PATCH	/mobile/api/v1/users/me{format}
DELETE	/mobile/api/v1/users/me{format}
POST	/mobile/api/v1/users/register/
POST	/mobile/api/v1/users/register{format}
POST	/mobile/api/v1/users/reset-password/
POST	/mobile/api/v1/users/reset-password{format}

The mobile user endpoints are necessary in order to perform crucial functionalities for privacy and security purposes.

2.3.2 Web User Endpoints

2.3.2.1 Web Alert Endpoints

GET	/web_admin/api/v1/alerts/
POST	/web_admin/api/v1/alerts/
GET	/web_admin/api/v1/alerts/{format}
POST	/web_admin/api/v1/alerts/{format}
GET	/web_admin/api/v1/alerts/{id}/
PUT	/web_admin/api/v1/alerts/{id}/
PATCH	/web_admin/api/v1/alerts/{id}/
DELETE	/web_admin/api/v1/alerts/{id}/
POST	/web_admin/api/v1/alerts/{id}/deactivate/
POST	/web_admin/api/v1/alerts/{id}/deactivate{format}
GET	/web_admin/api/v1/alerts/{id}{format}
PUT	/web_admin/api/v1/alerts/{id}{format}
PATCH	/web_admin/api/v1/alerts/{id}{format}
DELETE	/web_admin/api/v1/alerts/{id}{format}

The web alert endpoints are used to manage alerts, as well as deactivating an active alert.

2.3.2.2 *Web Case Endpoints*

GET	/web_admin/api/v1/cases/
POST	/web_admin/api/v1/cases/
POST	/web_admin/api/v1/cases/upload_image/
POST	/web_admin/api/v1/cases/upload_image{format}
GET	/web_admin/api/v1/cases/{format}
POST	/web_admin/api/v1/cases/{format}
GET	/web_admin/api/v1/cases/{id}/
PUT	/web_admin/api/v1/cases/{id}/
PATCH	/web_admin/api/v1/cases/{id}/
DELETE	/web_admin/api/v1/cases/{id}/
POST	/web_admin/api/v1/cases/{id}/archive_case/
POST	/web_admin/api/v1/cases/{id}/archive_case{format}
GET	/web_admin/api/v1/cases/{id}/close_case/
GET	/web_admin/api/v1/cases/{id}/close_case{format}
GET	/web_admin/api/v1/cases/{id}{format}
PUT	/web_admin/api/v1/cases/{id}{format}
PATCH	/web_admin/api/v1/cases/{id}{format}
DELETE	/web_admin/api/v1/cases/{id}{format}

One of the most commonly used entity in the web application is the case functionalities. They are used to get, edit, delete, archive or close a case and to upload images regarding a case.

2.3.2.3 *Web Facility Endpoints*

GET	/web_admin/api/v1/facilities/
POST	/web_admin/api/v1/facilities/
GET	/web_admin/api/v1/facilities/{format}
POST	/web_admin/api/v1/facilities/{format}
GET	/web_admin/api/v1/facilities/{id}/
PUT	/web_admin/api/v1/facilities/{id}/
PATCH	/web_admin/api/v1/facilities/{id}/
DELETE	/web_admin/api/v1/facilities/{id}/
POST	/web_admin/api/v1/facilities/{id}/add_child_in_facility/
POST	/web_admin/api/v1/facilities/{id}/add_child_in_facility{format}
POST	/web_admin/api/v1/facilities/{id}/completeness/
POST	/web_admin/api/v1/facilities/{id}/completeness{format}
POST	/web_admin/api/v1/facilities/{id}/facility_assign_hfm/
POST	/web_admin/api/v1/facilities/{id}/facility_assign_hfm{format}
POST	/web_admin/api/v1/facilities/{id}/remove_child_from_facility/
POST	/web_admin/api/v1/facilities/{id}/remove_child_from_facility{format}
GET	/web_admin/api/v1/facilities/{id}{format}
PUT	/web_admin/api/v1/facilities/{id}{format}
PATCH	/web_admin/api/v1/facilities/{id}{format}
DELETE	/web_admin/api/v1/facilities/{id}{format}

The endpoints above describe the functionalities of a facility. In addition, we are able to signify the presence (or not) of minors under a hosting facility.

2.3.2.4 *Web Feedback Endpoints*

GET	/web_admin/api/v1/feedbacks/
POST	/web_admin/api/v1/feedbacks/
POST	/web_admin/api/v1/feedbacks/upload_image/
POST	/web_admin/api/v1/feedbacks/upload_image{format}
GET	/web_admin/api/v1/feedbacks/{format}
POST	/web_admin/api/v1/feedbacks/{format}
GET	/web_admin/api/v1/feedbacks/{id}/
PUT	/web_admin/api/v1/feedbacks/{id}/
PATCH	/web_admin/api/v1/feedbacks/{id}/
GET	/web_admin/api/v1/feedbacks/{id}{format}
PUT	/web_admin/api/v1/feedbacks/{id}{format}
PATCH	/web_admin/api/v1/feedbacks/{id}{format}

The web feedback endpoints give the ability to the user to upload an image of an evidence, as part of the rest feedback information.

2.3.2.5 *Web Organisation Endpoints*

POST	/web_admin/api/v1/organizations/
GET	/web_admin/api/v1/organizations/{format}
POST	/web_admin/api/v1/organizations/{format}
GET	/web_admin/api/v1/organizations/{id}/
PUT	/web_admin/api/v1/organizations/{id}/
PATCH	/web_admin/api/v1/organizations/{id}/
DELETE	/web_admin/api/v1/organizations/{id}/
POST	/web_admin/api/v1/organizations/{id}/completeness/
POST	/web_admin/api/v1/organizations/{id}/completeness{format}
POST	/web_admin/api/v1/organizations/{id}/get_users/
POST	/web_admin/api/v1/organizations/{id}/get_users{format}
POST	/web_admin/api/v1/organizations/{id}/remove_user_from_organization/
POST	/web_admin/api/v1/organizations/{id}/remove_user_from_organization{format}
GET	/web_admin/api/v1/organizations/{id}{format}
PUT	/web_admin/api/v1/organizations/{id}{format}
PATCH	/web_admin/api/v1/organizations/{id}{format}
DELETE	/web_admin/api/v1/organizations/{id}{format}

Using the Organisation endpoints above, apart from the basic API calls (list, create, read, update, delete) we are able to get the number of occupants living in the facilities of an organisation as well as removing a user from the organisation.

2.3.2.6 *Web User Endpoints*

GET	/web_admin/api/v1/users/
POST	/web_admin/api/v1/users/forgot-password/
POST	/web_admin/api/v1/users/forgot-password{format}
POST	/web_admin/api/v1/users/login/
POST	/web_admin/api/v1/users/login{format}
POST	/web_admin/api/v1/users/logout/
POST	/web_admin/api/v1/users/logout{format}
GET	/web_admin/api/v1/users/me/
PUT	/web_admin/api/v1/users/me/
PATCH	/web_admin/api/v1/users/me/
DELETE	/web_admin/api/v1/users/me/
GET	/web_admin/api/v1/users/me{format}
PUT	/web_admin/api/v1/users/me{format}
PATCH	/web_admin/api/v1/users/me{format}
DELETE	/web_admin/api/v1/users/me{format}
POST	/web_admin/api/v1/users/register/
POST	/web_admin/api/v1/users/register{format}
POST	/web_admin/api/v1/users/reset-password/
POST	/web_admin/api/v1/users/reset-password{format}
GET	/web_admin/api/v1/users/{format}

Using the web user endpoints, we perform actions to authenticate a user or register a new account.

2.4 **Mobile Application**

2.4.1 **Description**

The ChildRescue Mobile App incorporates the key features of the ChildRescue platform which are needed for the support of the general public in the various operations of the platform. The end users of the mobile app are the visitors (unregistered users), simple users (registered users) and volunteers (certified volunteers of the participating organisations). The mobile application provides all the necessary functionalities in order to support: 1) exchange of information about active cases alerts between the organisations and the general public which includes sharing public information about an active case and receiving feedback, 2) rescue teams alerting and management which includes real time communication in chatrooms with the volunteers, 3) task management of volunteers, 4) the end user's account management, as well as other functionalities.

The first version of the ChildRescue mobile app includes the android version of the mobile app for the visitors, the simple users and the volunteers. The final version will include all the extended functionalities of the mobile app in android and iOS for the visitors and the simple users and in android

for the volunteers. More specifically, the relevant functionalities and versions of both versions are provided in the following table (Table 2-3).

Table 2-3: ChildRescue Mobile App Versions

Version	User Roles	Supported framework	Core functionalities
Version 1.0	Visitor	Android	-Receive case alerts -Provide feedback
	Simple user	Android	-Receive case alerts -Provide feedback -User profile
	Volunteer	Android	-Receive case alerts -Provide feedback -User profile
Final version	Visitor	Android, iOS	-Receive case alerts -Provide feedback
	Simple user	Android, iOS	-Receive case alerts -Provide feedback -User profile
	Volunteer	Android	-Receive case alerts -Provide feedback -User profile

The ChildRescue Mobile App consists of the following technical components:

- **UI Creator:** Responsible for the dynamic creation of the user interface which is offered to the end user.
- **Data Handler:** Implements the core business logic of the Mobile app and it is responsible for the management of the requests to the Backend Server by calling the respective REST endpoint of the Backend API and the routing of the information in the Mobile app client.
- **Notifications Manager:** Responsible for the management of the various notifications (push notifications, alerts, notifications for messages etc.)
- **Geolocations Manager:** Responsible for the management of the geolocation services of the mobile app

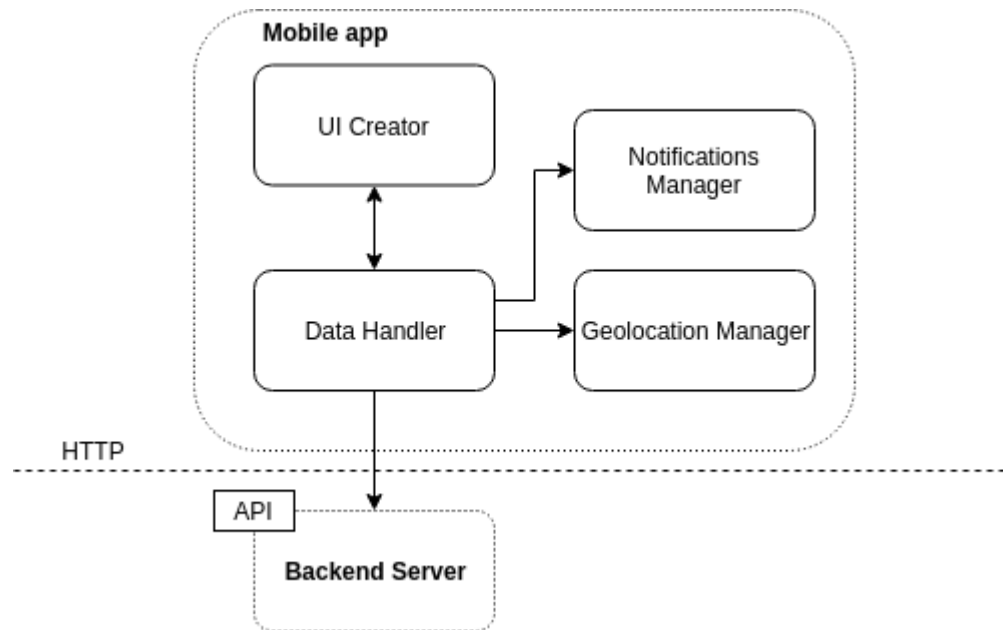


Figure 2-12: ChildRescue Mobile App Technical Components

The relevant technologies and frameworks that are being implemented by the aforementioned components are presented in the table below:

Table 2-4: ChildRescue Mobile App technologies and frameworks

Component	Implemented Framework
UI Creator	Android 8.0 Oreo ⁶
Data Handler	Android 8.0 Oreo
Notifications Manager	Firebase Cloud Messaging ⁷ , Android 8.0 Oreo
Geolocations Manager	LocationManager ⁸ , Geocoder ⁹ , Android 8.0 Oreo

2.4.2 Integration

The following sequence diagrams show the integration of the technical components of the ChildRescue Mobile App as well as the integration of the ChildRescue Mobile App with the Backend Platform. The sequence diagrams show the sequence of interactions between the aforementioned components based on three generic representative scenarios of a) request for content that is initiated from the Mobile App to the Backend Server, b) request for new alerts and c) receiving of new notifications.

⁶ <https://www.android.com/versions/oreo-8-0/>

⁷ <https://firebase.google.com/docs/cloud-messaging>

⁸ <https://developer.android.com/reference/android/location/LocationManager.html>

⁹ <https://developer.android.com/reference/android/location/Geocoder>

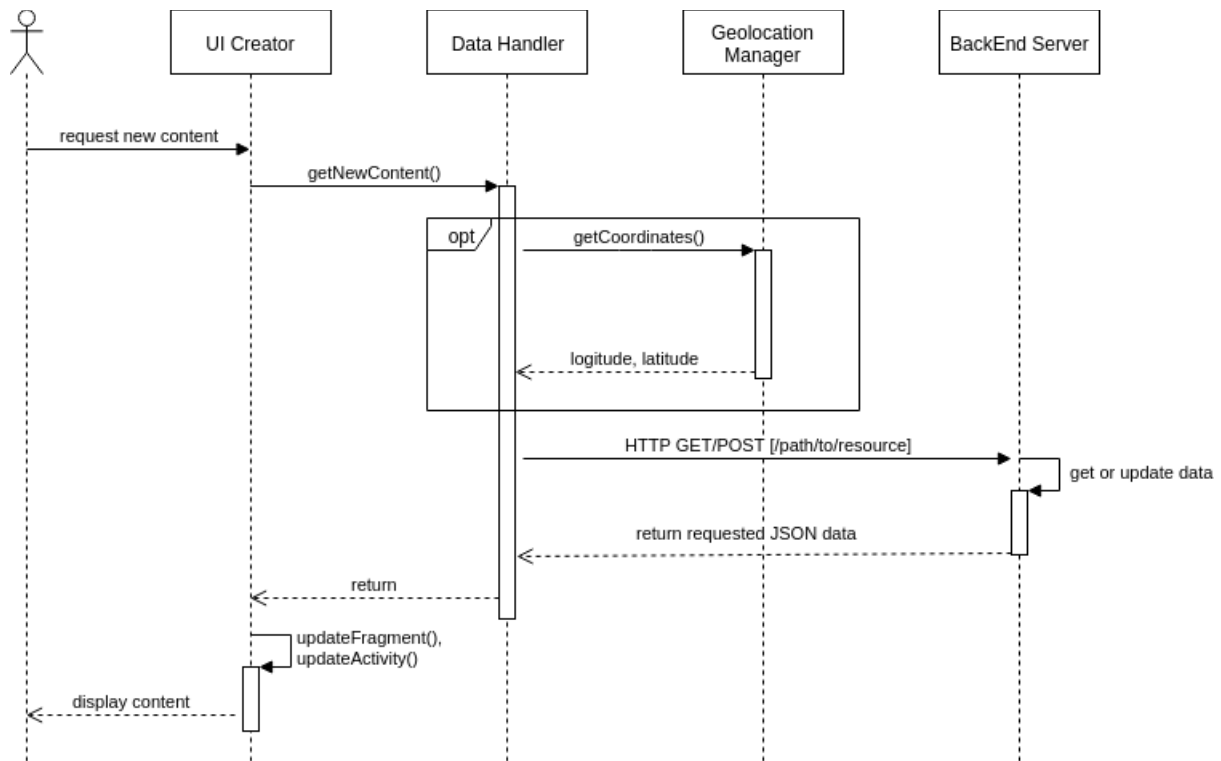


Figure 2-13: Mobile App Requests New Content

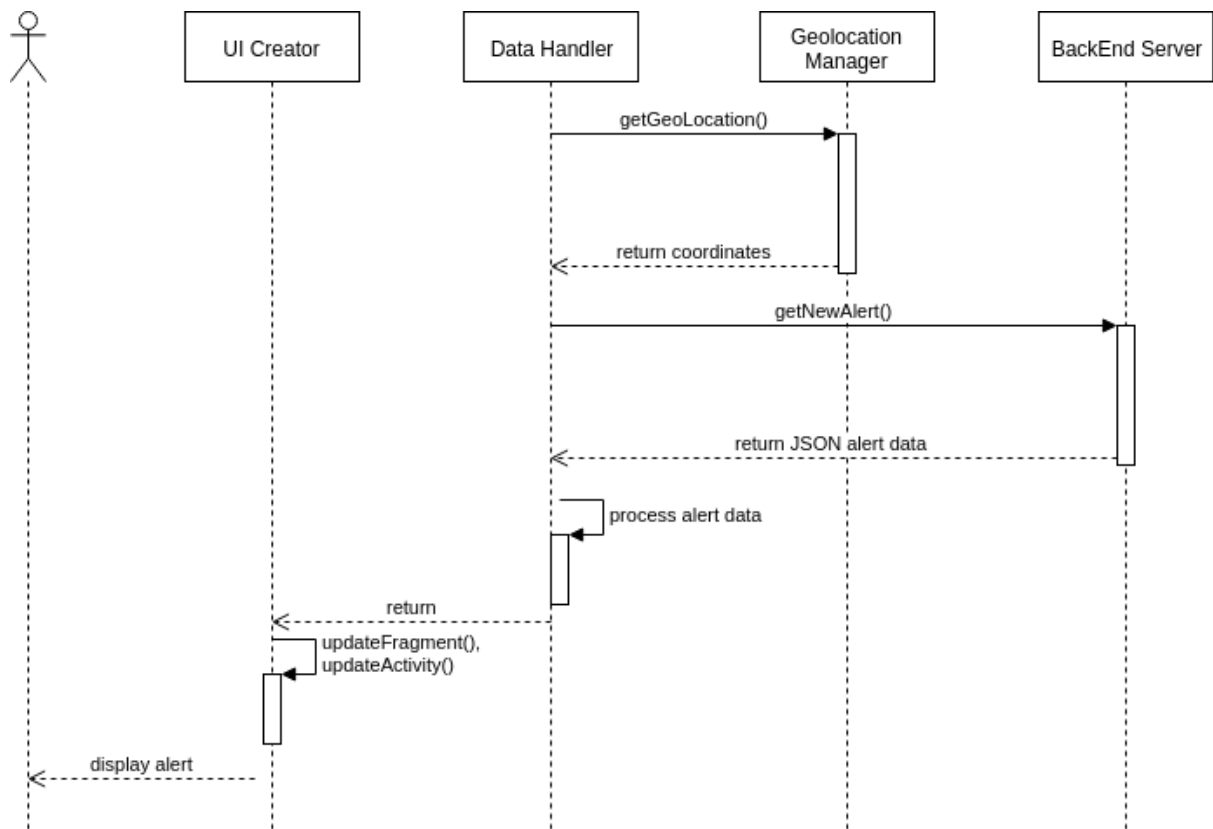


Figure 2-14: Mobile App Gets New Case Alert

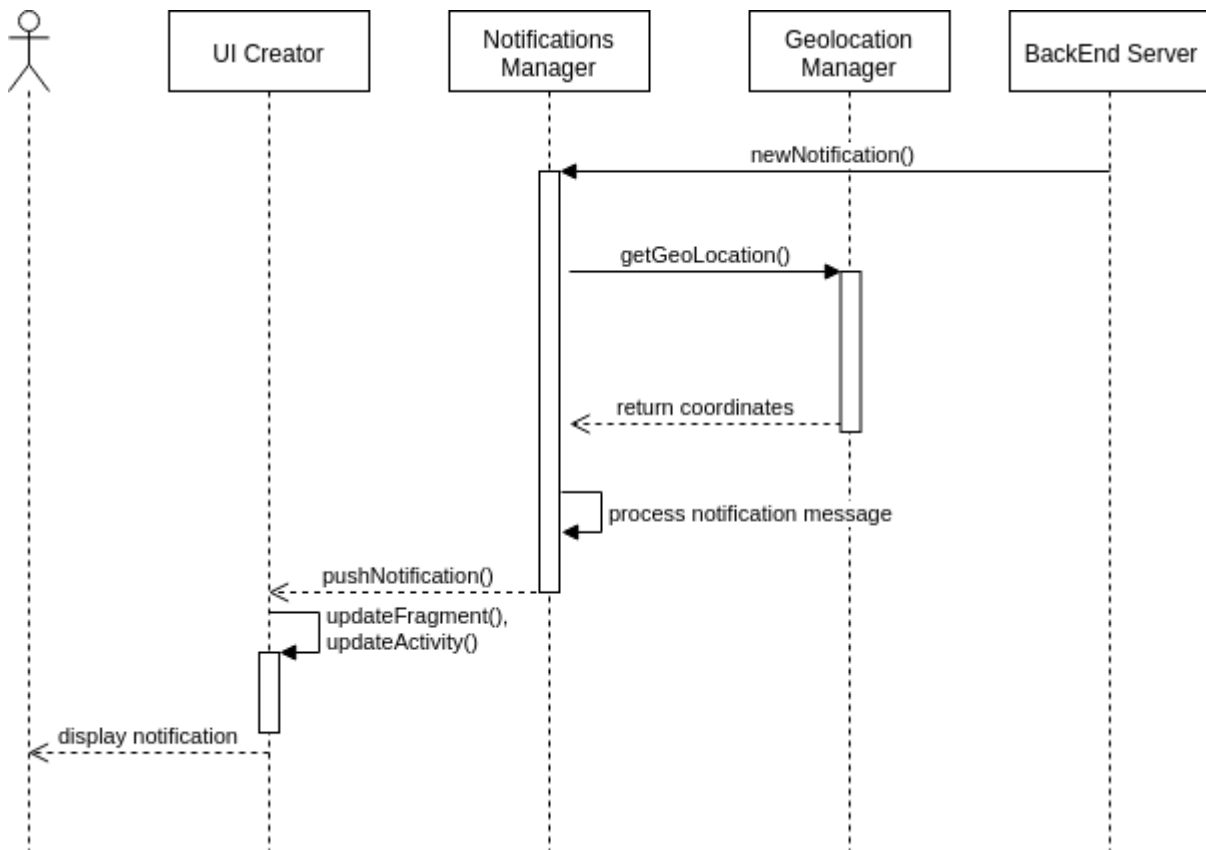


Figure 2-15: Mobile app Receives New Notification

2.4.3 Screenshots

In the current section, a set of representative screenshots of the mobile app for the simple users is presented since most of the features of v1.0 are common with the unregistered users and the volunteers. A detailed presentation of the mobile app is provided in "D4.2 – Pilots Planning and Preparation", in "Annex I: End User's Handbook".

2.4.3.1 Welcome

Upon installation two welcome screens are available. The first one (Figure 5) provides the title of the ChildRescue project and the second one (Figure 6) a brief description on how the user could get involved with ChildRescue. The user can proceed with the app either by pressing the "SKIP" button in the first case, or by pressing "GOT IT" in the second case.

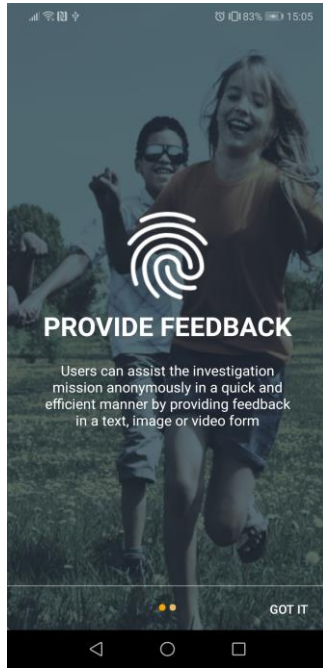


Figure 2-16: Welcome Description

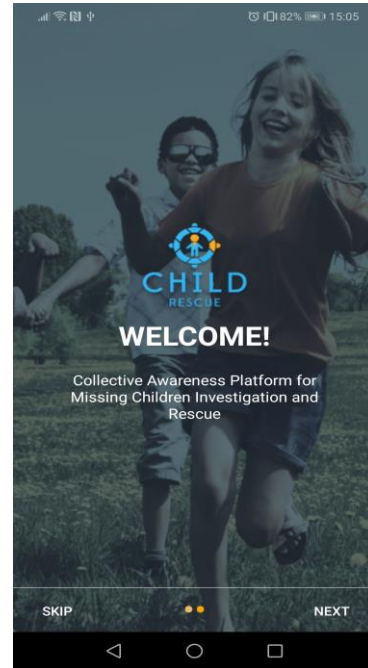


Figure 2-17: Welcome Title

2.4.3.2 *Login-Register*

As soon as the user visits the welcome page, she is redirected to the login screen (Figure 2-18), where she can choose whether she will login/register into the app or not.

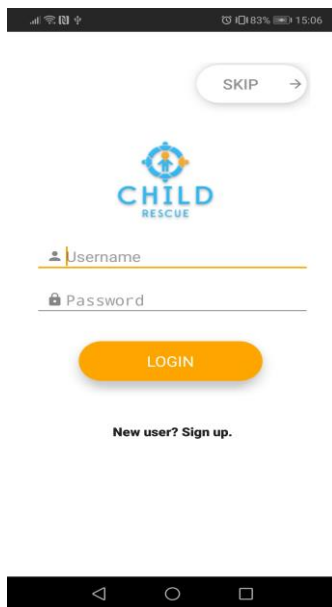


Figure 2-18: Login



Figure 2-19: Register Example

2.4.3.3 *Dashboard*

Once the user logs into the mobile app, she is provided with the main dashboard (Figure 2-20). The user is also able to access the details of her account.

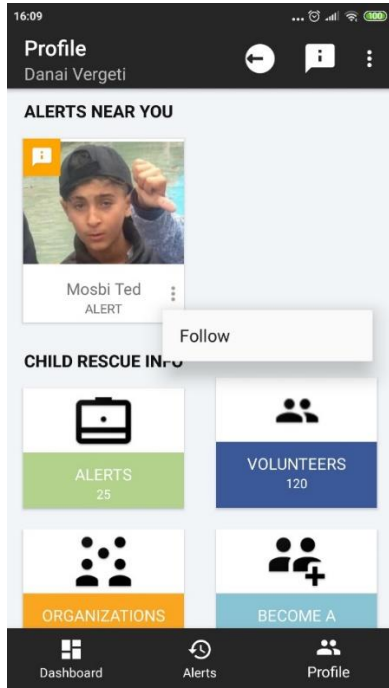


Figure 2-20: Dashboard - Registered User

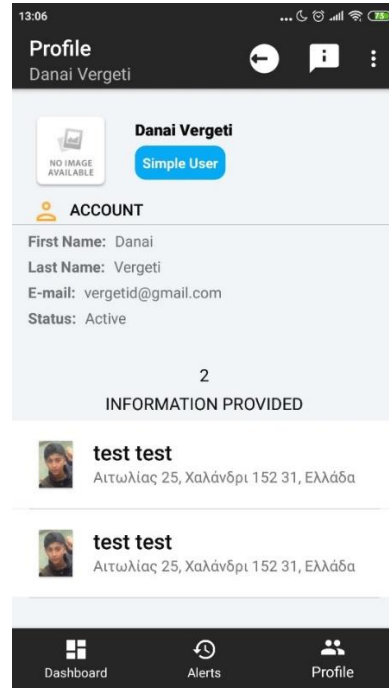


Figure 2-21: Profile - Registered User

2.4.3.4 *Provide Information for specific alert*

The user is able to provide feedback about a specific case by clicking on the alert card of the specific case on her dashboard (Figure 2-22).

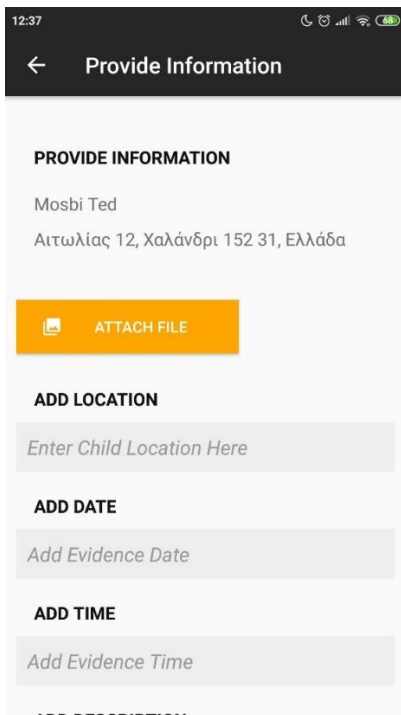


Figure 2-22: Provide Information - Complete Form (1/2)

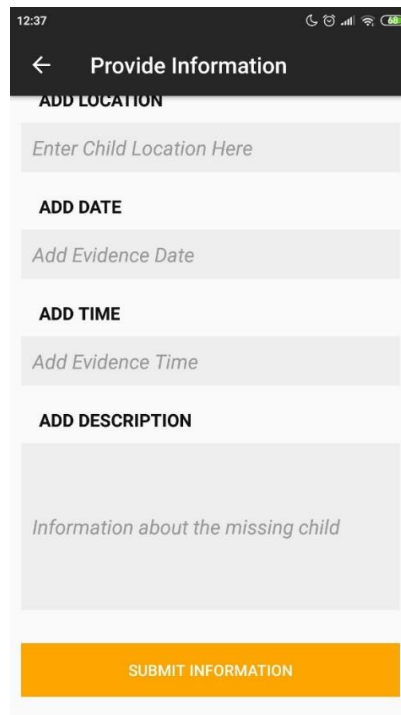


Figure 2-23: Provide Information - Complete Form (2/2)

2.4.3.5 Provide Information

Besides the general information that is available through the dashboard view, the user can provide information on her own, in case she wants to provide feedback related to a missing child case. To do so, she is redirected to the respective view (Figure 2-24) by pressing the “info” icon from the top bar as shown up in Figure 2-20.

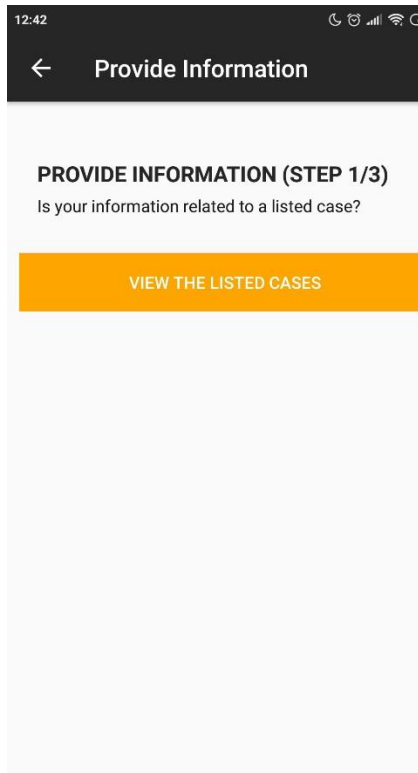


Figure 2-24: Provide Information Step 1

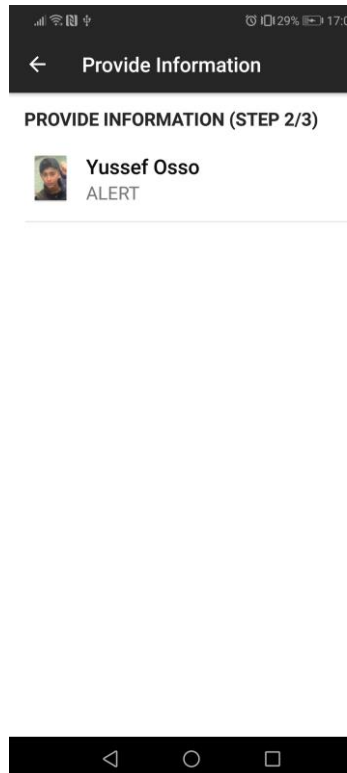


Figure 2-25: Provide Information Step 2

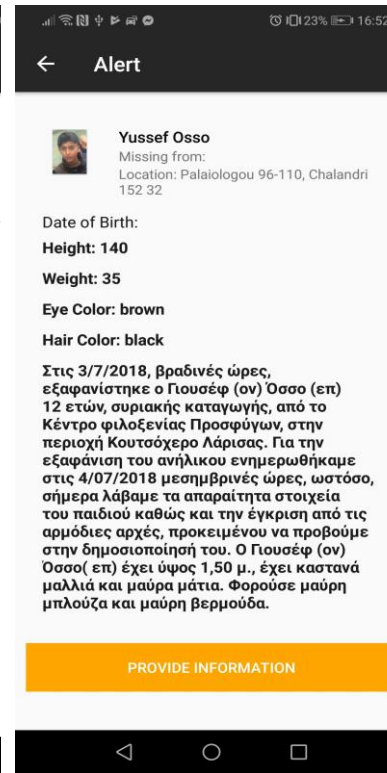


Figure 2-26: Provide Information

3 ChildRescue Integrated Platform Technical Verification and Evaluation

3.1 Technical Verification

A number of software projects rely solely on manual acceptance testing to verify that a piece of software conforms to its functional and non-functional requirements. Even where automated tests exist, they are often poorly maintained and out-of-date and thus require extensive manual testing. Testing is a cross-functional activity that involves the whole team, and should be done continuously from the beginning of the project. Building qualitative software implies the creation of automated tests at multiple levels (e.g. unit, integration and acceptance) and running them as part of the deployment pipeline which is triggered every time a change is made to the application, its configuration or the environment and software stack that it runs on. Manual testing is also an essential part of building quality in showcases, usability testing, and exploratory testing that need to be done continuously throughout the project. For the testing of the ChildRescue platform both automated and manual testing has been adopted in order to combine testing reliability of automated testing tools and also allow for human observation, which may be more useful if the goal is user-friendliness.

The current section presents the results of the testing of the ChildRescue platform which includes the testing results between the various integration points of the ChildRescue platform which assure the valid end-to-end operation of the platform.

3.1.1 Integration Testing

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group.

3.1.1.1 *Approaches and Techniques*

Integration testing is the extension of unit testing. The main idea of integration testing is to start from two units that have already been tested (and combined into a component) to test the interface between them. A component, in this sense, refers to an integrated aggregation of more than one unit. In a realistic scenario, many units are combined into components, which are in turn aggregated into even larger parts of the program. The idea is to test combinations of pieces and eventually expand the process to test the modules with those of other groups. Eventually all the modules making up a process are tested together. Beyond that, if the program is composed of more than one process, they should be tested **Error! Reference source not found.**

Integration testing identifies problems that occur when units are combined. By using a test plan that requires testing each unit and ensures the viability of each before combining units, any errors discovered when combining units are likely related to the interface between them. This method reduces the number of possibilities to a far simpler level of analysis.

A system expert can perform integration testing in a variety of ways but the following are three most common strategies:

- The top-down approach of integration testing requires the highest-level modules to be tested and integrated first. This allows high-level logic and data flow to be tested early in the process and it tends to minimize the need for drivers. However, the need for stubs complicates test

management and low-level utilities are tested relatively late in the development cycle. Another disadvantage of top-down integration testing is its poor support for early release of limited functionality.

- The bottom-up approach requires the lowest-level units to be tested and integrated first. These units are frequently referred to as utility modules. By using this approach, utility modules are tested early in the development process and the need for stubs is minimized. The downside, however, is that the need for drivers complicates test management and high-level logic and data flow are tested late. Like the top-down approach, the bottom-up approach also provides poor support for early release of limited functionality.
- The third approach, sometimes referred to as the "umbrella" approach, requires testing along functional data and control-flow paths. First, the inputs for functions are integrated in the bottom-up pattern discussed above. The outputs for each function are then integrated in the top-down manner. The primary advantage of this approach is the degree of support for early release of limited functionality. It also helps minimize the need for stubs and drivers. The potential weaknesses of this approach are significant, however, in that it can be less systematic than the other two approaches, leading to the need for more regression testing.

The ChildRescue consortium chose the last option, being the best, as it combines the best of both worlds. It allows all participating entities, to execute simultaneously multiple testing in several components. Integration testing plays a significant role for the proper functionality of the platform. As indicated by the system architecture in D3.1 the platform consists of a number of components implemented by different partners with different technologies addressing different functionalities. The valid integration of these components was a great need and integration testing became a priority at a very early stage of the development.

For testing the network communication of web services that were undergoing development we also employed service virtualisation techniques. In software engineering, service virtualisation is a method to emulate the behaviour of specific components in component-based applications. It is used to provide software development and testing teams access to dependent system components that are needed to exercise an application under test, but are unavailable or difficult-to-access for development and testing purposes. With the behaviour of the dependent components "virtualised" testing and development can proceed without accessing the actual live components, until they become available.

Service virtualisation involves creation and deployment of a "virtual asset" that simulates the behaviour of a real component which is required to exercise the application under test, but is difficult or impossible to access for development and testing purposes.

A virtual asset stands in for a dependent component by listening for requests and returning an appropriate response—with the appropriate performance. For example, in a web service, this involves listening for a RESTful message over HTTP and then returning another message. The virtual asset's functionality and performance tries to reflect the actual functionality of the component, also tries to simulate exceptional conditions (such as extreme loads or error conditions) to determine how the application under test responds under those circumstances.

Virtual assets are typically created either by providing logs representing historical communication among components, analysing service interface specifications (such as RESTful) or defining the behaviour manually with various interface controls and data source values. They are then further configured to represent specific data, functionality, and response times. After the release of a web

service prototype, it substitutes the virtualised one, in order for the integrated testing of the real one to be initiated.

3.1.2 Integration Points

In the following section we present a set of representative testing results of the integration between the backend platform and the UI as well as with the mobile app. The selected integration points offer the almost a full coverage of the functionalities provided in the first release of the ChildRescue platform.

3.1.2.1 User Registration

The relevant integration steps and interactions for the user registration are shown in the sequence diagram below:

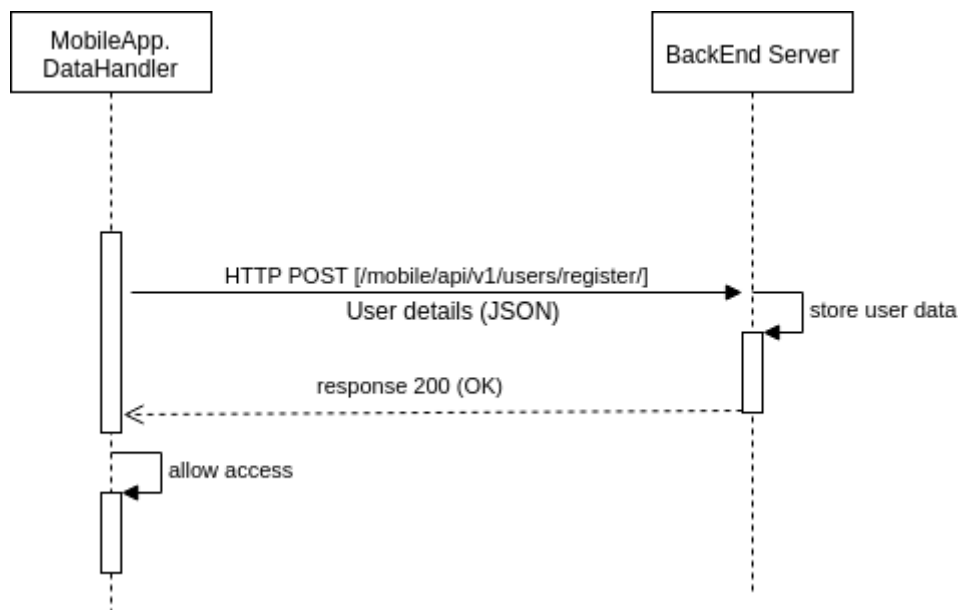


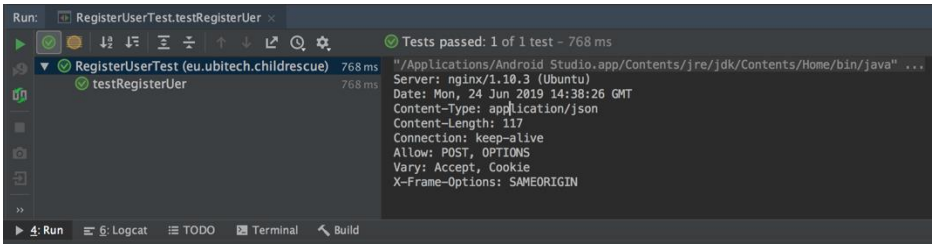
Figure 3-1: User Registration Integration Point

The relevant integration test is described below:

Table 3-1: User Registration Integration Test

Component	MobileApp.DataHandler
REST endpoint	http://157.230.102.159:8082/mobile/api/v1/users/register/
Method	POST
Caller	Call<User> createUser(@Field("first_name") String name, @Field("last_name") String lastName, @Field("email") String email, @Field("password") String password)
Unit test	<pre> package eu.ubitech.childrescue; import org.junit.Test; import java.io.IOException; import eu.ubitech.childrescue.entity.User; </pre>

	<pre> import eu.ubitech.childrescue.rest.ChildRescueAPIClient; import eu.ubitech.childrescue.rest.IChildRescueAPI; import retrofit2.Call; import retrofit2.Response; import static org.junit.Assert.assertTrue; public class RegisterUserTest { @Test public void testRegisterUer() { IChildRescueAPI apiService = ChildRescueAPIClient.getClient("").create(IChildRescueAPI.class); String firstName = "test"; String lastName = "user"; String email = "usertest@test.com"; String password = "user"; Call<User> call = apiService.createUser(firstName, lastName, email, password); try { Response<User> response = call.execute(); User userResponse = response.body(); assertTrue(response.isSuccessful() && userResponse != null); System.out.println(response.headers()); } catch (IOException e) { e.printStackTrace(); } } } </pre>
Request headers	<pre> Request originalRequest = chain.request(); Request.Builder builder = originalRequest.newBuilder() .header("Accept", "application/json") .header("Content-type", "application/json") .header("Authorization", "Bearer " + authorizationToken) .method(originalRequest.method(), originalRequest.body()); </pre>
Valid JSON Input sample	<pre> { "first_name": "test", "last_name": "user", "email": "usertest@test.com", </pre>

	<pre>"password": "user" }</pre>
Response headers	<pre>Server: nginx/1.10.3 (Ubuntu) Date: Mon, 24 Jun 2019 14:38:26 GMT Content-Type: application/json Content-Length: 117 Connection: keep-alive Allow: POST, OPTIONS Vary: Accept, Cookie X-Frame-Options: SAMEORIGIN</pre>
Response body	<pre>{ "id": 25, "email": "usertest@test.com", "first_name": "test", "last_name": "user", "profile_image": null, "is_end_user": false }</pre>
Execution results	

3.1.2.2 Receive Alert for specific case

The relevant integration steps and interactions for the available alerts are shown in the sequence diagram below:

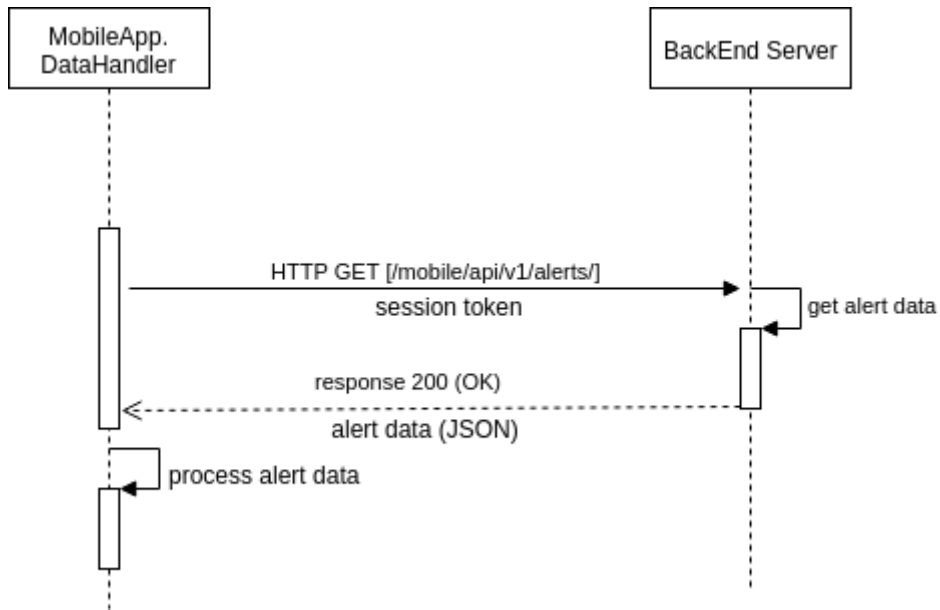


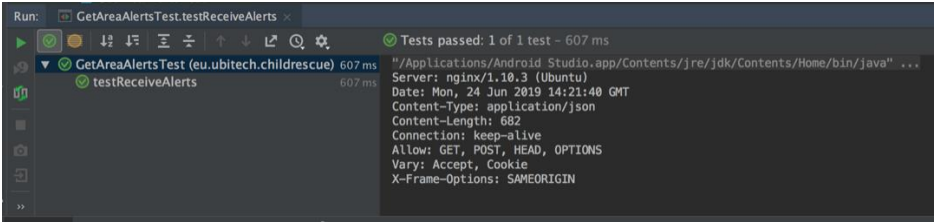
Figure 3-2 : Receive Alert Integration Point

The relevant integration test is described below:

Table 3-2: Receive Alert Integration Test

Component	MobileApp.DataHandler
REST endpoint	http://157.230.102.159:8082/mobile/api/v1/alerts/
Method	GET
Caller	Call<List<Alert>> fetchAreaAlerts(@Query("latitude") String latitude, @Query("longitude") String longitude);
Unit test	<pre> package eu.ubitech.childrescue; import org.junit.Test; import static org.junit.Assert.*; import java.io.IOException; import java.util.List; import eu.ubitech.childrescue.entity.Alert; import eu.ubitech.childrescue.rest.ChildRescueAPIClient; import eu.ubitech.childrescue.rest.IChildRescueAPI; import retrofit2.Call; import retrofit2.Response; public class GetAreaAlertsTest { @Test public void testReceiveAlerts() { </pre>

	<pre> IChildRescueAPI apiService=ChildRescueAPIClient.getClient("").create(IChildRescueAPI.class); //38.024813 //23.848919 String latitude = "38.024813"; String longitude = "23.848919"; Call<List<Alert>> call = apiService.fetchAreaAlerts(latitude, longitude); try { Response<List<Alert>> response = call.execute(); List<Alert> listResponse = response.body(); assertTrue(response.isSuccessful() && !listResponse.isEmpty()); System.out.println(response.headers()); } catch (IOException e) { e.printStackTrace(); } } } </pre>
Request headers	<pre> Request originalRequest = chain.request(); Request.Builder builder = originalRequest.newBuilder() .header("Accept", "application/json") .header("Content-type", "application/json") .header("Authorization", "Bearer " + authorizationToken) .method(originalRequest.method(), originalRequest.body()); </pre>
Valid JSON Input sample	<pre> { "latitude": "38.024813", "longitude": "23.848919" } </pre>
Response headers	<pre> Server: nginx/1.10.3 (Ubuntu) Date: Mon, 24 Jun 2019 14:21:40 GMT Content-Type: application/json Content-Length: 682 Connection: keep-alive Allow: GET, POST, HEAD, OPTIONS Vary: Accept, Cookie X-Frame-Options: SAMEORIGIN </pre>

Response body	<pre>[{ "id": 27, "case": 1, "geolocation_point": "SRID=4326;POINT (38.00773829999999 23.796386299999999)", "latitude": 38.0077383, "longitude": 23.7963863, "address": "Αιτωλίας 12, Χαλάνδρι 152 31, Ελλάδα", "radius": 30.0, "start": "2019-06-18T09:08:12.496000Z", "end": "2019-06-28T09:08:12.496000Z", "is_active": true, "description": "Missing child with red clothes", "created_at": "2019-06-18T09:08:12.678256Z", "updated_at": "2019-06-24T07:13:07.881966Z", "fullname": "Mosbi Ted", "disappearance_date": "2019-03-05T17:25:33Z", "date_of_birth": null, "eye_color": "brown", "hair_color": "brown", "height": 150, "weight": 41, "image": "/media/profile_images/p1chl2ps9d2g8q0o16tghp212ac4_400_D06F02C3.jpg" }]</pre>
Execution results	 <p>The screenshot shows the execution results of a test named <code>GetAreaAlertsTest.testReceiveAlerts</code>. The test passed successfully in 607 ms. The output includes the following details:</p> <ul style="list-style-type: none">Tests passed: 1 of 1 test - 607 msServer: nginx/1.10.3 (Ubuntu)Date: Mon, 24 Jun 2019 14:21:40 GMTContent-Type: application/jsonContent-Length: 682Connection: keep-aliveAllow: GET, POST, HEAD, OPTIONSVary: Accept, CookieX-Frame-Options: SAMEORIGIN

3.1.2.3 *Send feedback for specific case*

The relevant integration steps and interactions for the feedback provided by the user are shown in the sequence diagram below:

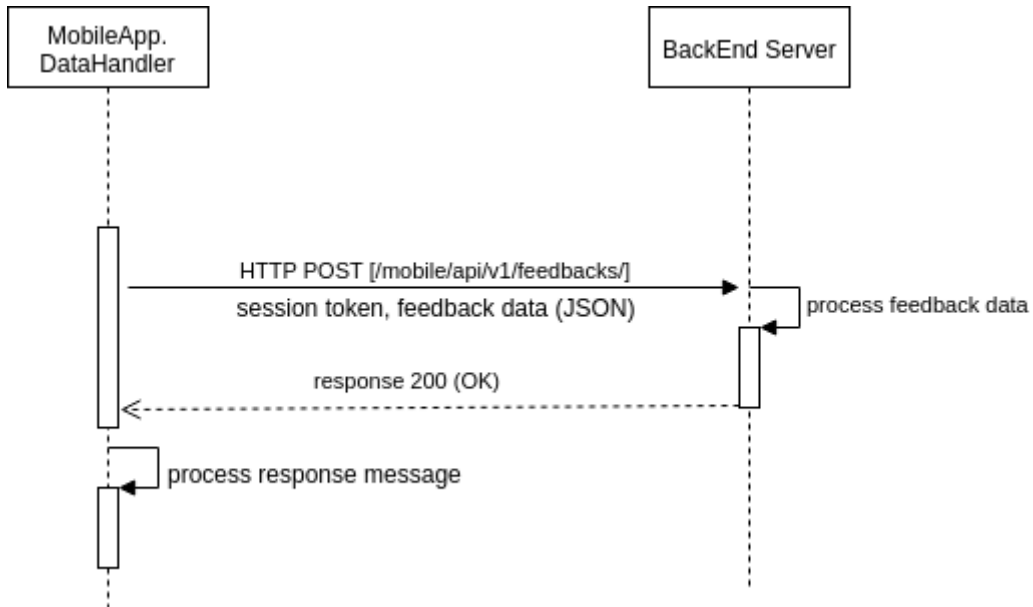


Figure 3-3: Send Feedback Integration Point

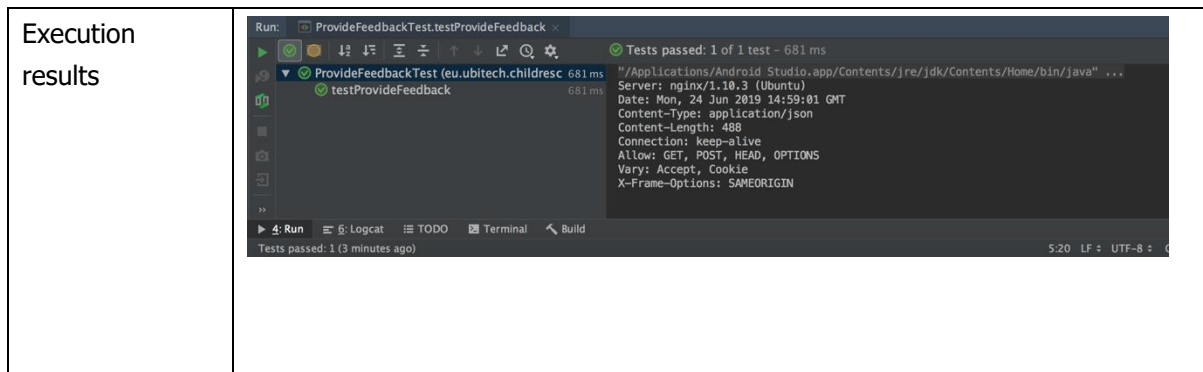
The relevant integration test is described below:

Table 3-3: Send Feedback Integration Test

Component	MobileApp.DataHandler
REST endpoint	http://157.230.102.159:8082/mobile/api/v1/feedbacks/
Method	POST
Caller	Call<Case> provideInfo(@Field("user") Integer userId, @Field("case") Integer caseId, @Field("current_longitude") String userLong, @Field("current_latitude") String userLat, @Field("longitude") String caseLong, @Field("latitude") String caseLat, @Field("comment") String description, @Field("date") String date);
Unit test	<pre> package eu.ubitech.childrescue; import org.junit.Test; import java.io.IOException; import eu.ubitech.childrescue.entity.Case; import eu.ubitech.childrescue.rest.ChildRescueAPIClient; import eu.ubitech.childrescue.rest.IChildRescueAPI; import retrofit2.Call; import retrofit2.Response; import static org.junit.Assert.assertTrue; public class ProvideFeedbackTest { </pre>

	<pre> @Test public void testProvideFeedback() { IChildRescueAPI apiService = ChildRescueAPIClient.getClient("").create(IChildRescueAPI.class); Integer userId = 1; Integer caseId = 1; String current_latitude = "38.024813"; String current_longitude = "23.848919"; String case_latitude = "38.024813"; String case_longitude = "23.848919"; String description = "test"; String date = "2019-06-12T20:00:00"; Call<Case> call = apiService.provideInfo(userId, caseId, current_longitude, current_latitude, case_longitude, case_latitude, description, date); try { Response<Case> response = call.execute(); Case feedbackResponse = response.body(); assertTrue(response.isSuccessful() && feedbackResponse != null); System.out.println(response.headers()); } catch (IOException e) { e.printStackTrace(); } } </pre>
Request headers	<pre> Request originalRequest = chain.request(); Request.Builder builder = originalRequest.newBuilder() .header("Accept", "application/json") .header("Content-type", "application/json") .header("Authorization", "Bearer " + authorizationToken) .method(originalRequest.method(), originalRequest.body()); </pre>
Valid JSON Input sample	<pre> { "userId":1, "caseId":1, "current_longitude":"23.848919", "current_latitude":"38.024813", "case_longitude":"23.848919", "case_latitude":"38.024813", "description":"test", </pre>

	<pre>"date": "2019-06-12T20:00:00" }</pre>
Response headers	<pre>Server: nginx/1.10.3 (Ubuntu) Date: Mon, 24 Jun 2019 14:59:01 GMT Content-Type: application/json Content-Length: 488 Connection: keep-alive Allow: GET, POST, HEAD, OPTIONS Vary: Accept, Cookie X-Frame-Options: SAMEORIGIN</pre>
Response body	<pre>{ "id": 97, "source": "Anonymous", "latitude": 38.024813, "longitude": 23.848919, "current_latitude": 38.024813, "current_longitude": 23.848919, "comment": "test", "feedback_image": null, "address": null, "date": "2019-06-12T20:00:00Z", "created_at": "2019-06-24T17:48:10.599333Z", "updated_at": "2019-06-24T17:48:10.599364Z", "checked_on": null, "feedback_status": "pending", "is_valid": null, "location_selected_reasons": null, "child_status": null, "transportation": null, "case": 1, "alert": null, "user": 1, "checked_by": null }</pre>



3.2 Technical Evaluation

For the technical evaluation of the ChildRescue platform, the “Product Quality Model” from ISO/IEC 25010:2011¹⁰ was used as a guidance for the generation of the technical KPIs of the Platform. Each aspect of the model was examined whether and how it covers the requirements of the project. Based on this analysis, a number of relevant KPIs were identified. Section 3.2.1 examines which of the Product Quality Model categories are relative to the project, while in section 3.2.2 the specific, project-related technical KPIs are described. This section describes solely the technical analysis, which took place at the latest cycle of the platform implementation.

3.2.1 Product Quality Model Categories

Table 3-4 showcases the sub-characteristics of each category of the Product Quality Model which are relevant to the ChildRescue platform.

Table 3-4: Product Quality Model Categories

Sub-characteristics	Definition	Relation to ChildRescue
Functional suitability		
Completeness	Degree to which the set of functions covers all the specified tasks and user objectives.	YES
Correctness	System provides the correct results with the needed degree of precision.	YES
Appropriateness	The functions facilitate the accomplishment of specified tasks and objectives.	YES
Performance efficiency		
Time behaviour	Response, processing times and throughput rates of a system, when performing its functions, meet requirements.	YES
Resource utilisation	The amounts and types of resources used by a system, when performing its functions, meet requirements.	YES
Compatibility		

¹⁰ <https://www.iso.org/standard/35733.html>

Co-existence	Product can perform its functions efficiently while sharing environment and resources with other products.	YES
Interoperability	A system can exchange information with other systems and use the information that has been exchanged.	YES
Usability		
Ease of Use	System has attributes that make it easy to operate and control.	YES
User error protection	System protects users against making errors.	YES
User interface aesthetics	User interface enables pleasing and satisfying interaction for the user.	YES
Technical Accessibility	System can be used by people with the widest range of characteristics and capabilities.	YES
Reliability		
Maturity	System meets needs for reliability under normal operation.	YES
Availability	System is operational and accessible when required for use.	YES
Fault tolerance	System operates as intended despite the presence of hardware or software faults.	YES
Security		
Confidentiality	System ensures that data is accessible only to those authorised to have access.	YES
Integrity	System prevents unauthorised access to, or modification of, computer programs or data.	YES
Non-repudiation	Actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.	YES
Authenticity	The identity of a subject or resource can be proved to be the one claimed.	YES
Maintainability		
Modularity	System is composed of components such that a change to one component has minimal impact on other components.	YES
Reusability	An asset can be used in more than one system, or in building other assets.	YES
Testability	Effectiveness and efficiency with which test criteria can be established for a system.	YES
Portability		
Installability	Effectiveness and efficiency with which a system can be successfully installed and/or uninstalled.	YES (only for mobile app)

To validate that the platform fulfils adequately all technical requirements, a set of technical methodologies and tools were used, and these include:

- W3C Mark-up Validation Service for HTML/XHTML and CSS validation in the context of technical quality assurance.
- Sonar for the integrated platform Quality Assurance
- Unit Testing on discrete platform components
- Integration testing on the integrated platform

3.2.2 Technical KPIs of the ChildRescue Platform

Based on the tables presented in the previous sections that reveal which criteria could be measured during the ChildRescue operation, and based on the fact that the ISO/IEC 25010:2011 standard does not define specific attributes (measuring ways) for each one of the sub-characteristics, the following list of KPIs has been defined to allow the technical evaluation of the ChildRescue platform.

Table 3-5: ChildRescue Technical Evaluation KPIs

Sub-characteristics	KPIs	Calculation Type	Recommended Limit	Value	Mandatory/Optional	Comments
Functional suitability						
Functional completeness	Percentage of completed Use Cases / Usage Scenarios	(Completed Use Cases / Defined Use Cases) * 100 %	100%	100 %	M	For v1.0 a set of use cases are executed. Final version will cover all use cases
Functional correctness	Percentage of Use Cases without reported bugs, after tests	(Completed Use Cases without bugs / Defined Use Cases) * 100 %	>90%	95%	M	No critical issues for v1.0. To be further analysed during the execution of the pilots.
Functional appropriateness	Percentage of tasks accomplished / Tasks Defined	(Accomplished Tasks / Tasks Defined) * 100%	>90%	100 %	M	No critical issues for v1.0. To be further analysed during the execution of the pilots for final version
Performance efficiency						
Time behaviour	Average Latency	(Total Response Time)/(No. of Requests)	<= 1 sec	<1 sec.	M	All requests are executed in 0,5 -1 sec.
Resource utilisation	Mean % CPU Utilisation	(Σ (% CPU utilisation probes))/(No. of probes)	<60%	<12 %	M	No critical issues for v1.0. To be further analysed in final version
	Mean Memory Usage	(Σ (RAM Megabytes used	<60%	<27 %	M	No critical issues for v1.0. To be

		in each probe)/(No. of probes)				further analysed in final version
Compatibility						
Co-existence	Ability to host in a single environment	Can the ChildRescue components operate in a shared environment? YES/NO	YES	YES	O	Virtualization and dockerization techniques are applied
Usability						
Ease of Use¹¹	Number of clicks required to reach requested information	Number of clicks required to reach requested information	<3	~2-3 clicks	M	
User error protection	System crash on user errors	Does the whole crash on user errors? YES/NO	NO	NO	M	
User interface aesthetics	User Interface Accessibility standardisation	No. of User Interface Accessibility standards followed	1	1	M	
Technical Accessibility	Cross-Platform Accessibility	ChildRescue platform is accessible and operational through different platforms (e.g. Windows / MacOS / Linux)	YES	YES	M	Mobile app v1.0 is available only in android. The final version of the mobile app (for visitors and simple users) will be also available in iOS.
	Cross-Browser Accessibility	ChildRescue platform is accessible and operational through different browsers (e.g. Chrome / Firefox/EDGE)	YES	YES	M	
	Cross-Device Accessibility	ChildRescue platform	YES	YES	M	No critical issues for v1.0. To be

¹¹ Behavioral characteristics are part of the quality in use model, here only metrics that can be automated extracted are mentioned

		accessible and operational through different devices (e.g. PC / Laptop/ Tablet / Smartphone)				further analysed in final version for the volunteers and network managers
Reliability						
Maturity	Max. Concurrent Users Supported	No. of Max. Concurrent Users Recorded	>100 users	>100 users	M	Achieved in simulation through automated stress testing for v1.0
	Simultaneous Requests	No. of Simultaneous Requests (e.g. page open)	>75	>100	M	Achieved in simulation through automated stress testing for v1.0
Availability	% Monthly Availability	1- ((Downtime Minutes)/(Month Days*24*60))	>90%	>99%	M	Any downtime documented was intentional and only for infrastructure upgrade.
	Error Rate	(No. of Problematic Requests)/(Total Number of Requests)	<10%	~2%	M	No critical issues for v1.0.
Fault tolerance	Number of Software problems identified without affecting the platform	No. of Non-Critical Software Errors	<10	3	M	No critical issues for v1.0.
Security						
Confidentiality	Incidents of ownership changes and accessing prohibited information	No. of incidents recorded	0 (zero)	0	M	No critical issues for v1.0.
Integrity	Incidents of authentication mechanism breaches	No. of incidents recorded	0 (zero)	0	M	No critical issues for v1.0.
Authenticity	Level of User Authenticity	Can you identify whether a subject is the one it claims to be? YES/NO/Partially	YES	YES	M	Volunteers are certified organization members.

	Level of Data Resource Authenticity	Can you identify whether a resource is the one it claims to be? (or e.g. is it created by robots) YES/NO/Partially	YES	YES	O	Integration and exchange of information is encrypted and token protected.
Maintainability						
Modifiability	% of Update Effectiveness	(No. of updates preformed without noticing operational problems)/(No. of updates performed)	>75%	100%	M	All updates were performed smoothly for v1.0
Testability	Level of Testing	Are tests able to probe the ChildRescue platform behaviour? YES/NO/Partially	YES	YES	M	Unit, integration and acceptance tests are also supported for the technical testing of the platform.
Portability						
Adaptability	Mean No. of Errors per Installation	(No. of Total Errors recorded during Installations)/(Total No. of Installations)	<1	0	O	No installation errors were recorded for v1.

4 Next Steps

Release II [M27] is the next official software release and the final one. It will be documented in D3.4 – “ChildRescue Platform, APIs and Mobile App, Release II”. In terms of functionality, it will include analytics – behaviour and predictive analytics, progress tracking and monitoring, evidence evaluation - , field operation and team coordination support, blockchain implementation, case classification and intelligent search. The relevant components should be developed and achieve UI-level integration by [M27]. Of course, all functionalities supported in the first release will be refined and further developed until the final release. User roles both for the web and the mobile application, that were not implemented in Release I, will be included in the second release.

It should be noted, however, that the technical partners are considering also two intermediate releases (R1.1 [M21], R1.2 [M23]) in order to accommodate feedback and bug fixes, and prepare the ChildRescue platform and app to support the training sessions and trial scenarios of pilot partners.

A full public release is due for launching in [M29].

Annex I: Implemented User Stories Release I

#	Epic	ACTOR	GOAL/DESIRE	BENEFIT	Voting Result	Planned for Release
		As a...	I want ...	, so that...		
VU.00	Download	Visitor	to download the light ChildRescue app	I can be informed, receive alerts and provide feedback	new story	I
VU.01	Registration	Visitor	to be able to register using minimal details	I can be part of the ChildRescue platform.	14	I
VU.01.2	Platform Browsing	Visitor	to be able to use the app without registering	I can still receive alerts and contribute anonymously, without having a ChildRescue account	new story	I
VU.01.3	Platform Browsing	Visitor	to be able to allow geolocation services	I can receive alerts based on my current location.	16,33333333	I
VU.02	Platform Browsing	Visitor	to be able to view currently active alerts near my vicinity	I can actively participate in the investigation.	10	I
VU.04	Platform Browsing	Visitor	to be able to visit the page with more information on a single case, when I receive notification for this case	I can view more details concerning the case.	10	I
VU.05	Platform Browsing	Visitor	to be able to view a list of the participating organisations and their contact details	I can communicate with organisations by other means, such as telephone or postal services.	9	I

VU.05.1	Platform Browsing	Simple User	to get informed on the process to become a volunteer for a specific organisation	I can offer my help in a more official and permanent way	10,5	I
VU.06	Platform Browsing	Visitor	to be able to view information about the platform, terms of service and usage, etc.	I can be fully aware of what the platform is about and what requirements it involves.	15	I
VU.07	Platform Browsing	Visitor	to be able to view information about DOs and DON'Ts regarding missing children investigation.	I can assist the investigation in an appropriate for the child at risk manner, abiding by the law and respecting human rights.	14	I
VU.08	Investigation Coordination/ Communication	Visitor	to send geotagged (automatically, with the current location of my mobile) and timestamped feedback in the form of text and image to the organisation concerning a missing child	I can assist in the investigation mission.	9	I
VU.08.2	Investigation Coordination/ Communication	Visitor	to view a list with the emergency operating lines of each organisation, with a brief description, when I want to provide input regarding a child and I don't find a matching case in the list	I can assist also in cases that I am not actively informed about, in the most appropriate manner.	new story	I
VU.08.3	Investigation Coordination/ Communication	Visitor	to view a confirmation screen after I submit feedback	I know that the process has been successful	new story	I
SU.04	Profile Editing	Simple User	to be able to use my "right to be forgotten" under the GDPR, and permanently remove my account from the system	I am no longer a member of the ChildRescue platform and my personal information cannot be retrieved by any means.	20	I
SU.12.1	Investigation Coordination/ Communication	Simple User	to be able to view all feedback I have submitted through the ChildRescue app	I have an overview of my actions.	n/a	I
SU.15	Investigation Coordination/	Simple User	to get notified when there is a missing child alert in my vicinity	I can act on it and help the investigation.	11,66666667	I

	Communication					
SU.16.1	Investigation Coordination/ Communication	Simple User	to have the option to select and "follow" specific cases	I receive notification if case is closed	10	I
SU.17	Investigation Coordination/ Communication	Simple User	to get notified when a case I follow or had provided feedback for, is closed	I can stop any actions I have started.	14	I
FM.01.1	Investigation Profiling	Hosting Facility Manager	to match a new child profile with existing profiles using simple name search (surname, first name)	I can identify if a newcoming minor has been registered before or was in another facility before.	12	I
FM.02	Investigation Profiling	Hosting Facility Manager	to be able to fill in profiling information about an unaccompanied migrant minor residing in the premises	I can keep track of the minors residing in the facility I am responsible for	9	I
FM.04.1	Investigation Profiling	Hosting Facility Manager	to be able to edit the case info and the documents in the ChildRescue platform	I can easily update the record of a minor under my responsibility.	20	I
FM.04.2	Facility Management	Hosting Facility Manager	to be able to see a list of all children accommodated in a hosting facility	I can have a better overview.	n/a	I
CM.01	Investigation Profiling	Organisation Case Manager	to be able to fill in the basic information about a new missing child case	I can provide basic details on that case.	17,5	I
CM.01.2	Investigation Profiling	Organisation Case Manager	to be able to edit the case info and documents in the ChildRescue platform	I can easily find and make changes to the history of a child.	20	I
CM.01.3	Investigation Profiling	Organisation Case Manager	to have access to all cases <i>of my organisation</i>	information about the same case is not lost between different case managers and shifts.	8,16666667	I

CM.01.6	Investigation Profiling	Organisation Case Manager	to be able to find a case that is documented in the system using simple name search	I can see if it is a multiple case and a relevant record already exists in the system.	new story	I
CM.02	Investigation Profiling	Organisation Case Manager	to update and extend the information of a case with more details or specify crucial information (like location last seen, latest info gathered, etc.)	I can offer an updated description of a case.	17,5	I
CM.02.1	Investigation Profiling	Organisation Case Manager	to update the open case record to with received feedback, after I have approved this information	all case data is stored in a well-maintained and continuous manner.	17,5	I
CM.02.2	Investigation Profiling	Organisation Case Manager	to be able to tag received feedback as irrelevant/relevant/credible	I can easily assess feedback.	n/a	II
CM.11.1	Investigation Crowd Sourcing Action	Organisation Case Manager	to be able to configure the centre, radius and duration of the alert for the crowdsourcing feedback validation process	I can focus the process.	13,41666667	I
CM.13.1	Investigation Crowd Sourcing Action	Organisation Case Manager	all notifications sent through the platform to the general public to include only information approved by the authorities	the whereabouts of the missing child cannot be inferred and malicious use of the platform is prevented.	14	I
CM.13.2	Investigation Crowd Sourcing Action	Organisation Case Manager	to configure the dissemination area and duration of the alerts that will be sent to users based on my expertise	crowdsourcing is more focused and efficient.	12	I
CM.13.3	Investigation Crowd Sourcing Action	Organisation Case Manager	to be able to change the configurations of an active alert or even deactivate it	I can have control on the alert system.	new story	I
CM.13.4	Investigation Crowd Sourcing Action	Organisation Case Manager	the alert to remain on the user's mobile until it expires or is set inactive by me	the location of the alert is not easily inferred.	new story	I

CM.15	Investigation Coordination/ Communication	Organisation Case Manager	to be able to get real-time feedback from all available human resources regarding a missing child alert (e.g. citizens, anonymous users)	I can gather all possible information	14	I
CM.18	Investigation Monitoring	Organisation Case Manager	to view all information about a case, sent by all participants, in historical order	I can have a more detailed view on the progress of the investigation	18,66666667	I
CM.23	Archiving	Organisation Case Manager	the personal case data of a closed case to not be retrievable any more through ChildRescue for any user, if the child has requested it	the rights of the child on its personal data, under GDPR, are respected.	new story	I
OC.01	Investigation Profiling	Organisation Coordinator	To have access to all cases' information and be able to edit, open or close a case	I have total control of all cases.	14	I
OC.06.1	Investigation Monitoring	Organisation Coordinator	to have a basic monitoring dashboard for each case	I can understand at which state is each case and what assistance is required.	new story	I
OO.01.1	Organisation Information Filling	Organisation Owner	to be able to set and edit the basic information about my organisation	I can provide an informative profile of the organisation.	20	I
OO.04.1	Organisation Management	Organisation Owner	to create profiles in the ChildRescue platform for administrative users of my organisation (OC, CM, FM, NM)	I can add admins of my organisation to the platform	new story	I
CA.01	ChildRescue Administration	ChildRescue Administrator	to create the organisation profile and add organisation owner	the organisation can use ChildRescue.	16,33333333	I
CA.02	ChildRescue Administration	ChildRescue Administrator	to delete an organisation	it is no longer part of the ChildRescue platform	15,16666667	I