



S H A P E S

Smart and Healthy Ageing through People Engaging in supporting Systems

D 4.3 – Integration Plan and Test Cases

| | |
|---------------------------|-------------------------------------------------------------------------------|
| Project Title | Smart and Healthy Ageing through People Engaging in Supportive Systems |
| Acronym | SHAPES |
| Grant Number | 857159 |
| Type of instrument | Innovation Action |
| Topic | DT-TDS-01-2019 |
| Starting date | 01/11/2019 |
| Duration | 48 |

| | |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Work package | WP4 – SHAPES Technological Platform |
| Lead author | Eleni ZAROGIANNI (ICOM) and Ilia PIETRI (ICOM) |
| Contributors | ICOM: Artur KRUKOWSKI EDGE : Marco Manso (EDGE), Jose Pires, Barbara Guerra GNO: Fotis Gonidis & Alexander Berler FINT: George Bogdos & Anargyros Sideris HMU: Yannis Nikoloudakis TREE: Jorge Fontela MedSyn: Christoph Kokelmann OMN: Waihang Shek SciFY: Paul Isaris UCLM: Felix Jesus Villanueva-Molina VICOM: Luis Unzueta, Manex Serras, Gorka Epelde, Jordi Torres, Naiara Muro, Jon Kerexata, Garazi Artola, Gorka Epelde & Eduardo Carrasco |
| Version | 1.0 |
| Due date | 31/10/2022 (M24) |
| Submission date | 31/10/2021 (M24) |

| | |
|----------------------------|--------------------------------|
| Dissemination Level | PU Public dissemination |
|----------------------------|--------------------------------|

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.



Revision History

Table 1 – Revision History

| Rev. # | Date | Editor | Comments |
|--------|----------|-----------------------------------------------------------|----------------------------------------------------------------|
| 0.0 | 15/4/20 | A. Krukowski (ICOM) | Initial ToC content |
| 0.1 | 7/8/21 | E. Zarogianni & I. Pietri (ICOM) | Deployment & testing |
| 0.2 | 7/9/21 | A. Krukowski (ICOM) | Integrated ToC |
| 0.3 | 20/9/21 | E. Zarogianni (ICOM) | Section 3.2.1.1, 3.2.1.2 & 3.2.1.3 |
| 0.4 | 21/9/21 | Y. Nikoloudakis (HMU) | Section 3.2.6 & 3.2.8 |
| 0.5 | 28/9/21 | J. Fontela (TREE) L. Unzueta (VICOM) | Sections 3.2.5, 3.2.7 & Annex 1 (1.2 & 1.3) |
| 0.6 | 29/9/21 | E. Zarogianni (ICOM) | Sections 1.3, 1.6, 2.1, 2.2, 2.3, 2.4, 2.6 & 2.7 |
| 0.7 | 4/10/21 | F.J. Villanueva-Molina (UCLM) | Annex 1 (1.4) |
| 0.7 | 4/10/21 | W. Shek (OMN) | Annex 1 (1.1) |
| 0.8 | 5/10/21 | M. Manso, J. Pires & B. Guerra (EDGE) | Section 3.2.7 |
| 0.9 | 6/10/21 | A. Sideris & G. Bogdos (FINT) | Sections 3.2.2 & 3.2.3 |
| 1.0 | 7/10/21 | I. Pietri & E. Zarogianni (ICOM) | Chapter 1, 2 & 3. Sections 1.2, 2.4, 2.6, 3.2.1.2 & 3.2.1.3 |
| 1.0. | 7/10/21 | Kokelmann C. (MedSyn) | Annex 1 (1.5) & Annex 2. |
| 1.1 | 8/10/21 | F. Gonidis (GNO) | Sections 3.2.4, 3.2.4.1-3.2.4.3 |
| 1.2 | 12/10/21 | P. Isaris (SciFY) I. Pietri & E.Zarogianni (ICOM) | Annex 2. Sections 1.3, 3.1 & 3.2.1 |
| 1.3 | 13/10/21 | I. Pietri & E.Zarogianni (ICOM) | All sections, Annex 1 & 2 |
| 1.4 | 15/10/21 | X.Garcia (EDGE) J.Fontela (TREE) A.Krukowski (ICOM) | Section 3.2.7.2 Section 3.2.5.2 All sections |
| 1.5 | 27/10/21 | E.Zarogianni, A.Krukowski | Amendments after peer reviews |

Keywords

Technological Platform, Design, Architecture, Implementation

Disclaimer

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

This document contains information which is proprietary to the SHAPES consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to any third party, in whole or parts, except with the prior written consent of the SHAPES coordinator.

Table of Contents

| | | |
|-----------|--------------------------------------------------------------|----------|
| 1 | <i>Introduction and Methodologies</i> | 1 |
| 1.1 | Partners Involved in Relevant Tasks in WP4 | 1 |
| 1.2 | Field of Application | 2 |
| 1.3 | Methodology for Test Case Deployment | 2 |
| 1.4 | Structure and Scope of the Document | 3 |
| 1.5 | Relation to other work in the project | 3 |
| 2 | <i>Assembly and Integration Plan</i> | 5 |
| 2.1 | Tool chain | 5 |
| 2.2 | Version control | 5 |
| 2.3 | Code repository | 5 |
| 2.4 | Continuous Integration Server | 6 |
| 2.5 | Component's integration | 6 |
| 2.6 | Lab deployment | 7 |
| 3 | <i>Integration testing of the software components</i> | 8 |
| 3.1 | Testing methodology | 8 |
| 3.2 | Components | 8 |
| 3.2.1 | The symbloTe orchestration middleware (ICOM) | 8 |
| 3.2.1.1 | Overview of deployment options | 8 |
| 3.2.1.2 | Interfaces (I/F) offered | 9 |
| 3.2.1.3 | Test cases and Validation | 10 |
| 3.2.2 | Gateway (FINT) | 15 |
| 3.2.2.1 | Overview of deployment options | 15 |
| 3.2.2.2 | Interfaces offered | 15 |
| 3.2.2.3 | Test cases and Validation | 16 |
| 3.2.3 | FiNoT IoT platform (FINT) | 16 |
| 3.2.3.1 | Overview of deployment options | 16 |
| 3.2.3.2 | Interfaces offered | 17 |
| 3.2.3.3 | Test cases and Validation | 18 |
| 3.2.3.3.1 | Update JWT token test | 19 |
| 3.2.3.3.2 | List FiNoT objects test | 20 |
| 3.2.3.3.3 | Get FiNoT object test | 21 |
| 3.2.3.3.4 | Get FiNoT object historical data test | 22 |
| 3.2.3.3.5 | Create FiNoT object test | 23 |
| 3.2.3.3.6 | Update FiNoT object test | 24 |
| 3.2.3.3.7 | Delete FiNoT object test | 25 |
| 3.2.4 | FHIR Medical Interoperability (GNO) | 26 |
| 3.2.4.1 | Overview of deployment options | 26 |
| 3.2.4.2 | Interfaces offered | 27 |
| 3.2.4.3 | Test cases and Validation | 27 |
| 3.2.5 | Big Data Platform: Data Lakehouse & Analytics Engine (TREE) | 27 |
| 3.2.5.1 | Overview of deployment options | 28 |
| 3.2.5.2 | Interfaces offered | 29 |
| 3.2.5.3 | Test cases and Validation | 34 |
| 3.2.6 | ASAPA Authentication and Authorisation (HMu) | 49 |

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.



| | | |
|----------------|------------------------------------------------------------------------------------------------------|-----------|
| 3.2.6.1 | Overview of deployment options | 50 |
| 3.2.6.2 | Interfaces offered | 50 |
| 3.2.6.3 | Test cases and Validation..... | 51 |
| 3.2.7 | Front-end App (EDGE)..... | 53 |
| 3.2.7.1 | Overview of deployment options | 53 |
| 3.2.7.2 | Interfaces offered | 54 |
| 3.2.7.3 | Test cases and Validation..... | 55 |
| 3.2.8 | Marketplace (HMU) | 59 |
| 3.2.8.1 | Overview of deployment options | 60 |
| 3.2.8.2 | Interfaces Offered..... | 60 |
| 3.2.8.3 | Test cases and Validation | 60 |
| 4 | Results and Conclusions | 70 |
| 5 | Ethical Requirements Check..... | 71 |
| 6 | References | 73 |
| Annex 1 | Digital Solutions – Integration Efforts..... | 74 |
| 1.1 | NOTIFY Digital Solution (OMN) | 74 |
| 1.1.1 | Overview of deployment options | 74 |
| 1.1.2 | Interfaces offered | 74 |
| 1.1.3 | Test cases and Validation..... | 76 |
| 1.2 | FACECOG Tool to Support User Authentication and for People Identification at a Distance (VICOM)..... | 77 |
| 1.2.1 | Overview of deployment options | 77 |
| 1.2.2 | Interfaces offered | 79 |
| 1.2.3 | Test cases and Validation..... | 84 |
| 1.3 | Adilib Chatbot Building Platform (VICOM)..... | 84 |
| 1.3.1 | Overview of deployment options | 84 |
| 1.3.2 | Interfaces offered | 85 |
| 1.3.3 | Test cases and Validation..... | 90 |
| 1.4 | At-home Rehabilitation System (UCLM) | 90 |
| 1.4.1 | Overview of deployment options | 91 |
| 1.4.2 | Interfaces offered | 91 |
| 1.4.3 | Test cases and Validation..... | 91 |
| 1.5 | MedicalSyn Digital Solution (MedSyn)..... | 97 |
| 1.5.1 | Overview of deployment and customization options | 97 |
| Annex 2 | Integration of Digital Solutions with ASAPA and Front-end App. | 99 |
| 2.1 | Conversion of needs to specifications for integration..... | 99 |
| 2.2 | Refactoring of a Digital Solution to accommodate ASAPA | 99 |
| 2.3 | Integrating Front-end App with a Digital Solution | 99 |

List of Tables

| | |
|-----------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| TABLE 1 – REVISION HISTORY | III |
| TABLE 2 – ACRONYMS AND ABBREVIATIONS. | XI |
| TABLE 3 – DELIVERABLE CONTRIBUTORS..... | 1 |
| TABLE 4 – SYMBIoTe API INTERFACES. | 9 |
| TABLE 5 - INTERFACE SYMBIoTeAPI-1 . THIS INTERFACE IS USED TO REGISTER AN EXISTING ASAPA USER TO SYMBIoTe. | 10 |
| TABLE 6 - INTERFACE SYMBIoTeAPI-2 . THIS INTERFACE IS USED TO REGISTER A NEW L1 RESOURCE TO SYMBIoTe. | 10 |
| TABLE 7- INTERFACE SYMBIoTeAPI-3 . THIS INTERFACE IS USED TO GET A LIST OF L1 RESOURCES. | 11 |
| TABLE 8 - INTERFACE SYMBIoTeAPI-4 . THIS INTERFACE IS USED TO GET RESOURCE INFORMATION FROM AN L1 RESOURCE USING ITS NAME. | 11 |
| TABLE 9 - INTERFACE SYMBIoTeAPI-5 . THIS INTERFACE IS USED TO GET RESOURCE INFORMATION FROM AN L1 RESOURCE USING ITS ID. | 12 |
| TABLE 10 - INTERFACE SYMBIoTeAPI-6 . THIS INTERFACE IS USED TO ACCESS AN L1 RESOURCE USING ITS NAME. | 13 |
| TABLE 11 - INTERFACE SYMBIoTeAPI-7 . THIS INTERFACE IS USED TO DELETE AN L1 RESOURCE. | 13 |
| TABLE 12 - INTERFACE SYMBIoTeAPI-8 . THIS INTERFACE IS USED TO RETRIEVE A LIST OF L2 RESOURCES..... | 14 |
| TABLE 13 - INTERFACE SYMBIoTeAPI-9 . THIS INTERFACE IS USED TO GET INFORMATION ABOUT AN L2 RESOURCE..... | 14 |
| TABLE 14 - INTERFACE SYMBIoTeAPI-10 . THIS INTERFACE IS USED TO ACCESS AN L2 RESOURCE. | 15 |
| TABLE 15 – SHAPES GW INTERFACES. | 16 |
| TABLE 16 - INTERFACE GWAPI . THIS INTERFACE IS USED TO CHECK THE GW STATUS..... | 16 |
| TABLE 17- FINOT’S PLATFORM INTERFACES | 17 |
| TABLE 18 - INTERFACE FINOTAPI . THIS INTERFACE IS USED TO AUTHENTICATE AN EXISTING USER TO FINOT. . | 18 |
| TABLE 19 - INTERFACE FINOTAPI . THIS INTERFACE IS USED TO REFRESH THE JWT AUTHENTICATION TOKEN. .. | 19 |
| TABLE 20 - INTERFACE FINOTAPI . THIS INTERFACE IS USED TO LIST ALL OF SHAPES OBJECTS (I.E., IOT DATA) HOSTED IN FINOT. | 20 |
| TABLE 21 - INTERFACE FINOTAPI . THIS INTERFACE IS USED TO GET A SPECIFIC SHAPES OBJECT (I.E., IOT DATA) HOSTED IN FINOT. | 21 |
| TABLE 22 - INTERFACE FINOTAPI . THIS INTERFACE IS USED TO GET A SPECIFIC SHAPES OBJECT (I.E., IOT DATA) HISTORICAL DATA HOSTED IN FINOT..... | 22 |
| TABLE 23 - INTERFACE FINOTAPI . THIS INTERFACE IS USED TO CREATE A SHAPES OBJECT (I.E., IOT DATA) IN FINOT..... | 23 |
| TABLE 24- INTERFACE FINOTAPI . THIS INTERFACE IS USED TO UPDATE A SHAPES OBJECT (I.E., IOT DATA) IN FINOT..... | 24 |
| TABLE 25 - INTERFACE FINOTAPI . THIS INTERFACE IS USED TO DELETE A SHAPES OBJECT (I.E., IOT DATA, CURRENT AND HISTORICAL) IN FINOT. | 26 |
| TABLE 26 – GNOMON’S INTERFACES..... | 27 |
| TABLE 27 – FHIR MQ TEST CASES | 27 |
| TABLE 28 - BIG DATA PLATFORM INBOUND API:..... | 30 |
| TABLE 29 - BIG DATA PLATFORM OUTBOUND API:..... | 31 |

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.



| | |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| TABLE 30 - CASE-INB-01. THIS TEST CASE CHECKS IF A NEW USER CAN BE REGISTERED TO THE INBOUND API SUCCESSFULLY..... | 34 |
| TABLE 31 - CASE-INB-02. THIS TEST CASE CHECKS IF A NEW USER CAN BE LOGGED IN TO THE INBOUND API SUCCESSFULLY..... | 35 |
| TABLE 32 - CASE-INB-03. THIS TEST CASE CHECKS IF A FHIR DATA JSON CAN BE UPLOADED TO THE INBOUND API SUCCESSFULLY..... | 35 |
| TABLE 33 - CASE-OUT-01. THIS TEST CASE CHECKS IF SLEEP ACTIVITY DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY. | 36 |
| TABLE 34 - CASE-OUT-02. THIS TEST CASE CHECKS IF SLEEP ACTIVITY DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY FOR A GIVEN USER_ID..... | 37 |
| TABLE 35 - CASE-OUT-03. THIS TEST CASE CHECKS IF SLEEP ACTIVITY DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY FOR A GIVEN A DATE RANGE. | 38 |
| TABLE 36 - CASE-OUT-04. THIS TEST CASE CHECKS IF SLEEP ACTIVITY DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY FOR A GIVEN USER_ID AND A DATE RANGE. | 38 |
| TABLE 37 - CASE-OUT-05. THIS TEST CASE CHECKS IF PHYSICAL ACTIVITY DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY. | 39 |
| TABLE 38 - CASE-OUT-06. THIS TEST CASE CHECKS IF PHYSICAL ACTIVITY DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY FOR A GIVEN USER_ID..... | 40 |
| TABLE 39 - CASE-OUT-07. THIS TEST CASE CHECKS IF PHYSICAL ACTIVITY DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY FOR GIVEN A DATE RANGE..... | 41 |
| TABLE 40 - CASE-OUT-08. THIS TEST CASE CHECKS IF PHYSICAL ACTIVITY DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY FOR A GIVEN USER_ID AND A DATE RANGE. | 42 |
| TABLE 41 - CASE-OUT-09. THIS TEST CASE CHECKS IF VITALS CONTROL CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY..... | 42 |
| TABLE 42 - CASE-OUT-10. THIS TEST CASE CHECKS IF VITALS CONTROL DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY FOR A GIVEN USER_ID..... | 43 |
| TABLE 43 - CASE-OUT-11. THIS TEST CASE CHECKS IF VITALS CONTROL CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY FOR A GIVEN A DATE RANGE..... | 44 |
| TABLE 44 - CASE-OUT-12. THIS TEST CASE CHECKS IF VITALS CONTROL DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY FOR A GIVEN USER_ID AND A DATE RANGE. | 45 |
| TABLE 45 - CASE-OUT-13. THIS TEST CASE CHECKS IF ANOMALY DETECTION DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY. | 46 |
| TABLE 46 - CASE-OUT-14 THIS TEST CASE CHECKS IF ANOMALY DETECTION DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY FOR A GIVEN USER_ID..... | 47 |
| TABLE 47 - CASE-OUT-15. THIS TEST CASE CHECKS IF ANOMALY DETECTION DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY FOR A GIVEN A DATE RANGE. | 48 |
| TABLE 48 - CASE-OUT-16. THIS TEST CASE CHECKS IF ANOMALY DETECTION DATA CAN BE FETCHED FROM THE OUTBOUND API SUCCESSFULLY FOR A GIVEN USER_ID AND A DATE RANGE. | 48 |
| TABLE 49 – ASAPA’S INTERFACES. | 50 |
| TABLE 50 - INTERFACE ASAPAAPI-1. THIS INTERFACE IS USED FOR CHECKING THE HEALTH OF ASAPA..... | 51 |
| TABLE 51 - INTERFACE ASAPAAPI-2. THIS INTERFACE IS USED TO REGISTER A USER TO ASAPA. | 51 |
| TABLE 52 - THE ENDPOINT OF BACKEND HEALTH. | 60 |
| TABLE 53 – COMPLIANCE CHECK ON ETHICAL REQUIREMENTS. | 71 |
| TABLE 54 – NOTIFY INTERFACES. | 75 |

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.



| | |
|-------------------------------------------------------------------------------|----|
| TABLE 55 – NOTIFY: REQUEST DATA FROM THE SERVER..... | 75 |
| TABLE 56 – OMNITORAPI TEST CASES. | 76 |
| TABLE 57 - ASUMMARY OF THE TESTING STEPS FOR THE UCLM DIGITAL SOLUTION: | 96 |

List of Figures

| | |
|----------------------------------------------------------------------------------------------|----|
| FIGURE 1- ANDROID STUDIO AVD MANAGER..... | 54 |
| FIGURE 2 - SHAPES FRONT-END APP LOGIN SCREEN | 56 |
| FIGURE 3 - SUCCESSFUL LOGIN IN THE SHAPES FRONT-END APP | 57 |
| FIGURE 4 - SHAPES FRONT-END APP MAIN SCREEN | 58 |
| FIGURE 5 – SHAPES MARKETPLACE LOGIN PAGE. | 61 |
| FIGURE 6 – MARKETPLACE’S HOMEPAGE AFTER SUCCESSFUL LOGIN. | 62 |
| FIGURE 7 – MARKETPLACE PRODUCT LIST | 62 |
| FIGURE 8 – MARKETPLACE: CREATE NEW PRODUCT. | 63 |
| FIGURE 9 – MARKETPLACE PRODUCT LIST..... | 63 |
| FIGURE 10 – MARKETPLACE’S PRODUCT INFORMATION PAGE. | 64 |
| FIGURE 11 – MARKETPLACE: ADDITIONAL PRODUCTION INFORMATION DETAILS. | 65 |
| FIGURE 12 - MARKETPLACE: PRODUCT PURCHASE SCENARIO (STEP 1). | 65 |
| FIGURE 13 - MARKETPLACE: PRODUCT PURCHASE SCENARIO (STEP 2). | 66 |
| FIGURE 14 - MARKETPLACE: PRODUCT PURCHASE SCENARIO (STEP 3). | 66 |
| FIGURE 15 – BEFORE FILLING PROFILE DETAILS. | 67 |
| FIGURE 16 - AFTER FILLING PROFILE DETAILS. | 67 |
| FIGURE 17 – VERIFY STATUS..... | 68 |
| FIGURE 18 – MARKETPLACE CHECKOUT PAGE. | 68 |
| FIGURE 19 – SHAPES MARKETPLACE HOMEPAGE AFTER SUCCESSFUL REGISTRATION OF A NEW PRODUCT. | 69 |
| FIGURE 20 - DEPLOYMENT DIAGRAM OF FACECOG. | 78 |
| FIGURE 21 – ADILIB’S WEB-BASED INTERFACE..... | 86 |
| FIGURE 22 – WEBCHAT WIDGET..... | 86 |
| FIGURE 23 – DEMO CHATBOT. | 90 |
| FIGURE 24 – LEGEND USED IN TEST MONITORING PIPELINE. | 92 |
| FIGURE 25 – ADMIN ACTIONS DEVELOPED AND TESTED. | 92 |
| FIGURE 26 – MANAGER ACTIONS DEVELOPED AND TESTED..... | 93 |
| FIGURE 27 – FACILITY, TOTEM AND EVENT ACTIONS DEVELOPED AND TESTED..... | 93 |
| FIGURE 28 – END-USER ACTIONS DEVELOPED AND TESTED. | 94 |
| FIGURE 29 – TRAINER ACTIONS DEVELOPED AND TESTED..... | 94 |
| FIGURE 30 – ROUTINE AND EXERCISE ACTION IMPLEMENTED AND TESTED. | 95 |

Table of Acronyms and Abbreviations

Table 2 – Acronyms and Abbreviations.

| Acronym | Description |
|---------------|----------------------------------------------------------------------|
| AIV | Assembly, Integration and Verifications |
| API | Application Programming Interface |
| App | Application |
| ASAPA | Authentication, Security and Privacy Assurance |
| CO | Consortium only dissemination level |
| PU | Public Dissemination level |
| DoA | Description of the Action |
| EC | European Commission |
| FHIR | Fast Healthcare Interoperability Resources |
| ICD | Interface Control Document |
| GDPR | General Data Protection Regulation |
| JSON | JavaScript Object Notation |
| MedSyn | MedicalSyn GmbH |
| OMN | Omnitor AB |
| PM | Person Month |
| QA | Quality Assurance |
| RAMS | Reliability, Availability, Maintainability, Safety of Means & People |
| REST | REpresentational State Transfer |
| RIA | Research and Innovation Action |
| SciFY | Science for You |
| STC | Scientific Technical Coordinator |
| TP | Technological Platform |
| TRD | Technical Requirements Document |
| UCLM | Universidad de Castilla - La Mancha |
| URL | Uniform Resource Locator |
| VICOM | Vicomtech |
| WP | Work Package |

Executive Summary

The deliverable D4.3 “Integration Plan and Test Cases” reports on the first phases of work produced in Task 4.8. “*Integration and Testing of SHAPES Technological Platform*” with respect to planned actions related to integration and deployments of the SHAPES platform for use case evaluations. As such it has also included contributions from all partners involved in WP4 tasks related to the implementation of relevant technical solutions as well as Digital Solutions developed in WP5 and Pilot Themes in WP6.

The structure of the document is as follows:

Section 1 introduces the methodologies and technologies implemented and concerning the integration of the SHAPES core components into the SHAPES core platform and outlines the scope and structure of the document.

Section 2 described the overall assembly, integration and testing plan of core components, as well as organisation of reference code repository. It also identifies approaches used for both lab and release types of deployments.

Section 3 describes deployment, interfaces, and integration and validation tests of the core components. Each component owner describes the main functionality of the component in question, its deployment options and interfaces, and finally the test cases for testing their components’ interfaces.

Section 4 summarises the outcomes of Task 4.8 and state of Technological Platform integration and testing, demonstrating its readiness for subsequent integration with Digital Solutions in view of further deployment in Pilot Themes.

Section 5 contains an obligatory Ethical Self-Check table as requested by WP8.

Appendices describe integration of Digital Solutions with the most important and essential core components of the SHAPES core Technological Platform, those being ASAPA (authentication system being a pre-requisite for being able to access any of the core components within the SHAPES-TP) and the Front-end application (offering single point of access for Digital Solutions and the SHAPES-TP).

1 Introduction and Methodologies

Deliverable D4.3 “Integration Plan and Test Cases” describes the plan for the integration of the software components into the SHAPES core platform and testing of their proper functioning as part of the integrated prototype. An extensive set of test cases for each component implemented in the development tasks is documented to validate that the interfaces of each component which is an outcome of the work in WP4 are well-functioning.

1.1 Partners Involved in Relevant Tasks in WP4

The main task producing this deliverable was Task 4.8 (Integration and Testing of SHAPES TP) that focussed on the integration and validation of the integrated SHAPES technological platform. However, since not all involved partners were involved in this specific task, we’ve asked additional partners like EDGE and VICOM to contribute with respect to their individual core components as part of their effort in Tasks 4.3 (Implementation of the Mediation Framework and Interoperability Services), Task 4.5 (Human Interaction and Visual Mapping) and Task 4.7 (Task 4.7: SHAPES Gateway Reference Implementation). A complete list of consortium partners contributing to deliverable D4.3 was as follows:

Table 3– Deliverable Contributors.

| ID | Short Name | Role |
|----|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15 | ICOM | Leader of WP4, Tasks 4.1, 4.2, 4.3 and 4.8 and deliverables D4.1 and D4.3. ICOM has led integration of the whole D4.3 deliverable and to specific sections 1, 2 introductory part of 3, and 3.1 related to symbloTe. |
| 8 | EDGE | Contributed to section 3.2.7 related to the Frontend App and Annex 2 related to digital solutions integration in mobile devices. |
| 12 | FINT | As a leader of Task 4.7, FINT contributed to specific sections 3.2.2 & 3.2.3 on IoT and gateway components |
| 13 | GNO | It contributed to sections 3.2.4, 3.2.4.1-3.2.4.3 directly related to their FHIR interoperability component |
| 16 | KOM | Supported work in Task 4.8 with respect to offline access to SHAPES-TP from their robots |
| 18 | MedSyn | Its contributions are in Annex 1 (1.5) and Annex 2 |
| 20 | OMN | Its contribution can be found in Annex 1 (1.1) |
| 22 | PAL | Supported work in Task 4.8 with respect to offline access to SHAPES-TP from their robots |

| | | |
|----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26 | SciFy | Its contribution is provided in Annex 2 |
| 27 | HMU | As a leader of Task 4.6, it contributed specifically to sections 3.2.6 & 3.2.8, related to its ASAPA component |
| 28 | TREE | As a leader of Task 4.4, it contributed specifically to sections 3.2.5 & 3.2.7, referring to its Datalake and Analytics engine components in a core of SHAPES-TP |
| 29 | UCLM | Its contribution can be found in Annex 1 (1.4) |
| 35 | VICOM | As a leader of Task 4.5, it contributed specifically to Annex 1 (1.2 & 1.3) |

1.2 Field of Application

This document is applicable to the remaining work within WP4 aiming at testing the integration of the core components into SHAPES TP, as well as the integration with the digital solutions that will be implemented in WP5. The various modules of the SHAPES TP will be integrated and tested in laboratory environment from a functional perspective. The integrated prototype will be ready to be deployed for the pilots and tested by the end users in accordance to the scenario uses cases.

1.3 Methodology for Test Case Deployment

The project consortium has followed the integration strategy, described in [1] for the implementation of the software components and their incorporation to the integrated SHAPES core platform. This deliverable describes in detail the methodology used towards the preparation of the integration activities, in order to successfully deliver the integrated software prototype in incremental releases.

A micro-service architecture has been followed to develop the SHAPES core platform, as a collection of loosely coupled components, the SHAPES core components, and enable their independent development, maintenance and deployment. A GitHub repository is available for maintaining the source code of all core components, and necessary information for building and deploying each component, such as the Docker files and the Docker compose files. Each software component is encapsulated in a container image and uploaded at Docker Hub, which is used as the Docker container repository for the purposes of the project.

The containerised components are deployed at the testing environment, that is the infrastructure that is set up to perform the integration activities and testing, using the provided configuration files. Each SHAPES core component provides a set of well-defined interfaces for its communication and inter-connection with other SHAPES core components.

Test cases for each of the components' interfaces and the relevant scripts required for their testing are provided by the respective SHAPES partners (component owners) to be used by the integrator, ICOM, in order to perform the integration testing. For

RESTful APIs, an HTTP request is sent to the relevant endpoint(s) to be tested and the response received is validated based on the expected result. In case of failures the relevant developers are informed to fix any errors. When all test cases of each SHAPES core component are successfully completed, the source code from the develop branch is pushed to the master branch

1.4 Structure and Scope of the Document

The document structure is composed of the following Sections:

- **Section 1:** Introduction and Methodologies
Describes the methodologies for the integration activities.
- **Section 2:** Assembly and Integration Plan
Lists the tools and implementation framework used for the deployment of core components and their integration into a software prototype.
- **Section 3:** Integration testing of the software components
Describes the testing of the software interfaces among the core components of the SHAPES core platform prototype. Reports deployment requirements, the API interfaces and the testing and validation efforts.
- **Section 4:** Results and Conclusions
- **Annexes:** provide additional information regarding the integration of digital solutions.
 - Annex 1: describes integration of NOT!FY (OMN), FACECOG (VICOM), Adilib Chatbot (VICOM), At-home Rehabilitation System (UCLM) and MedicalSyn (MedSyn) digital solutions to the SHAPES core platform.
 - Annex 2: describes the integration plan for digital solutions with the ASAPA and Front-end app components.

1.5 Relation to other work in the project

This deliverable is based on results from:

- **D4.1** “SHAPES TP Requirements and Architecture” (due M18)
- **D4.2** “SHAPES TP Development Tools and Capabilities Toolkit” (due M24)
- **D4.6** “SHAPES Interoperability Reference Testing Environment” (due M18)
- **D6.1** “SHAPES Pan-European Pilot Campaign Plan” (due M6)
- **D8.4** “Ethics Framework for Shapes solution” (due M6)

The outcomes from D4.3 feed to:

- **WP6:** “SHAPES Pan-European Pilot Campaign”

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

- **WP7:** *“Market Shaping, Scale-up Business Models and Socio-Economic Impact”*

2 Assembly and Integration Plan

This section describes the integration plan that is followed within the SHAPES project to ensure the successful delivery of the SHAPES core platform. It describes the implementation framework used for the development of the SHAPES core components and their integration into a working software prototype. It sets common methodologies, tools and workflows to be followed by all SHAPES developers in order to facilitate the development, deployment and testing of the integrated system.

2.1 Tool chain

GitHub and Docker tools will be used to facilitate the implementation workflow and support incremental feature deployment, maintenance and scalability of the highly distributed SHAPES ecosystem.

2.2 Version control

Git is an open-source distributed version control system that allows developers to collaborate, manage and track changes of the source code. It allows developers to work in parallel, by using a local copy of the repository at their working environment and committing their changes to it. When ready, local changes can be pushed onto a remote repository (the git server), where other developers can see the changes and pull them to update their local repository. Also, multiple local branches are supported that can enable the parallel development of features, as well as the management of product releases. Git will be used in the SHAPES project to enable its developers to work in parallel and collaborate on the source code for the development and maintenance of SHAPES core components. Two main branches are maintained to facilitate the integration and release process; a master branch pointing to the latest source code in production-ready state (i.e., latest release version) and a develop branch pointing to the latest development changes to be delivered for a next release.

2.3 Code repository

GitHub is a cloud-based Git repository hosting service for open-source projects. It will be used to host and make available the source code of the SHAPES project. GitHub public repositories and free accounts are used. More specifically, a remote GitHub repository has been set at: <https://github.com/SHAPES-H2020>.

To reflect the micro service architecture used, a multi-repository setup is used with the SHAPES core components being bundled into GitHub super-repositories. In that way, independent development of the software components can be achieved with clear ownership. Also, the build is faster due to the smaller code base, as developers need to download and build only the repositories they use. Project members are able to download the latest versions, fetch changes from other members, implement their features locally and commit their changes back to the shared repository. For each repository, a main contact is assigned administrator rights in order to be able to

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

manage their corresponding repository, create projects related to their component, add team developers, and organise their code base, as needed. Finally, GitHub provides access control and collaboration features such as bug tracking, feature requests and task management, which can be utilised during the development phase in the project.

2.4 Continuous Integration Server

Continuous integration is the practice of merging the source code changes frequently at a regular basis to develop and test the software in smaller increments. Travis CI¹ is a continuous integration server that supports the development process by enabling the automatic build and testing of code changes. It also allows to manage notifications and automatic deployment of the software. Commands for building, testing or deployment can be specified in a configuration file (*travis.yml*) in the correct code branch.

Travis is used for the purposes of the SHAPES project to automatically build the SHAPES core components software frequently. It has a smooth integration with GitHub in order to support the automatic software build and testing. A GitHub user has been setup for pushing code to the develop branch of the components. The URL where the SHAPES-H2020 organization can be accessed from in Travis is: <https://app.travis-ci.com/organizations/SHAPES-H2020>.

The Travis CI server is notified by GitHub, whenever a new commit is pushed to that repository or when a new pull request is submitted. Responsible developers are notified whenever new commits are pushed to that repository or a pull request is submitted.

2.5 Component's integration

RESTful APIs will be used to expose services provided by the SHAPES core components.

Swagger² is an open-source tool that provides a formal format to develop and document RESTful APIs. The OpenAPI Specification (currently in version 3.0.3³) defines a standard, language-agnostic interface to RESTful APIs in a way that both humans and computers can understand the RESTful services and their functionalities. It allows to describe the entire API including any available endpoints and operations on each endpoint (GET, POST etc.). Swagger is used in the context of the SHAPES project to define the RESTful APIs that SHAPES core components expose.

Finally, regarding the asynchronous communication between the SHAPES core components, a RabbitMQ server is deployed at ICOM's premises to enable the

¹ <https://www.travis-ci.com/>

² Swagger: <https://swagger.io/>

³ Open API spec v3.0.3: <https://swagger.io/specification>

exchange of notifications between SHAPES core components. Asynchronous APIs for the components that publish/write to a topic and “subscribe to”/“read from” a topic, are defined in the relevant components’ interfaces.

There are different serialization formats to be used for the transmission of data between applications. Among them, JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications, as it is human readable and lightweight and can be used for the SHAPES project needs.

2.6 Lab deployment

Docker is a software platform that allows the quick creation, testing and deployment of applications, by packaging software into standardized units called containers.

Each SHAPES core component is encapsulated in a container image to be deployed independently in the form of a Docker container. Component owners, the consortium partners that own SHAPES core components need to provide the configuration files required to build the Docker images for the deployment of their components, such as Docker files or Docker Compose files for more complex components.

As described in [1], Docker Hub⁴ is a service provided by Docker for discovering and sharing container images within a team. Docker Hub is used as the container image repository for the needs of the SHAPES project in order to store and share the Docker images required for the deployment and integration of the SHAPES core components. For this purpose, a Docker Hub account has been set up at the following URL: <https://hub.docker.com/u/shapes2020>. The Docker file of each SHAPES core component is used to automatically build the Docker images of the SHAPES core components within Travis CI and upload them to the Docker Hub repository. The container images uploaded at Docker Hub are then used to deploy the SHAPES core components at the testing infrastructure, using deployment YAML configuration files to perform the integration testing which is described in detail in the next section. When integration testing succeeds the new release is delivered.

⁴ Docker HUB: <https://hub.docker.com>



3 Integration testing of the software components

This section describes the testing of the software interfaces among the core components of the SHAPES core platform prototype. Reporting provides general information for each software component, the deployment requirements and options and the API interfaces provided for integration along with other SHAPES core components and their testing and validation.

3.1 Testing methodology

Integration testing is the phase in software testing where one or two individual software components are combined and tested as a group. It is used to test a service and its functionalities to validate the interactions between the subsystems and ensure that they work fine together, for example the database and the application. It is performed once the new functionalities of the components to be included in the new release are implemented, with the scope of detecting errors related to the components interfaces. In order to release the new version, all the integration tests defined for the testing of the software components of the system must be successful. Within the context of the SHAPES project, test cases for testing the SHAPES core components' interfaces and their functionalities are defined with the help of the component owners. The tests defined are then executed by the integrator (ICOM) to ensure the correct functioning of the interfaces and smooth interaction between the integrated components. In case of test failures, the respective SHAPES developers are notified to fix identified bugs to ensure the correct operation of the system to be released.

3.2 Components

3.2.1 The symbloTe orchestration middleware (ICOM)

The symbloTe is the IoT middleware of the SHAPES core platform, enabling the discovery and sharing of IoT health devices and services across IoT platforms, digital solutions or SHAPES core components, in a unified and secure manner. It allows the exchange of resource meta-information required to describe the IoT devices and services to be exposed, implementing a secure interworking protocol between the IoT platforms, gateways and smart devices.

3.2.1.1 Overview of deployment options

The symbloTe software can be deployed using Docker. The symbloTe core server acts as a centralised IoT search engine, where IoT platforms can register their resources and users (such as third-party applications) and search for these resources. The symbloTe server is deployed at ICOM's premises at the following URL: <https://symbiote-core.intracom-telecom.com/>, running the latest stable version.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

Additionally, a symbloTe API component has been developed in order to offer a set of REST services designed to cover basic client requirements without requiring applications or developers to download symbloTe libraries. The symbloTe API offers services for a user to interact with symbloTe, such as the registration of a valid (ASAPA) user to the symbloTe ecosystem, the registration, accessing and listing of resources (L1 or L2 compliance levels).

The symbloTe API has been developed and deployed in a Docker environment, running at the following URL: <http://146.124.106.199:8443>.

3.2.1.2 Interfaces (I/F) offered

Table 4– SymbloTe API interfaces.

| I/F | Feature | Endpoint/Queue | Description | Producer/ Resource | Consumer/ Caller |
|------|--------------------------|--------------------------------------------|-----------------------------------------------------------|-----------------------|---------------------------------|
| REST | POST – register user | /symbiote/admin/registerUser/me/toSymbiote | It registers an existing SHAPES user to symbloTe. | symbloTeAPI | Any other SHAPES core component |
| REST | POST – register resource | /symbiote/resource/register/L1Res | It registers an L1 resource to symbloTe. | symbloTeAPI | Any other SHAPES core component |
| REST | POST – get resource | /symbiote/resource/get/L1Res | It returns a list of L1 resources registered to symbloTe. | symbloTeAPI | Any other SHAPES core component |
| REST | POST – access resource | /symbiote/resource/access/sensor/L1Res | It provides access to L1 resource. | symbloTeAPI | Any other SHAPES core component |
| REST | POST – delete resource | /symbiote/resource/delete/L1Res | It deletes an L1 resource from symbloTe. | symbloTeAPI | Any other SHAPES core component |
| REST | POST – get L2 resource | /symbiote/resource/get/ListOfL2 | It returns a list of L2 Resources. | symbloTeAPI | Any other SHAPES core component |

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

| I/F | Feature | Endpoint/Queue | Description | Producer/Resource | Consumer/Caller |
|------|------------------------------|---------------------------------|--------------------------------------|-------------------|---------------------------------|
| REST | POST – info of L2 resource | /symbiote/resource/get/L2Res | It returns info about L2 Resources. | symbloTeAPI | Any other SHAPES core component |
| REST | POST – access an L2 resource | /symbiote/resource/access/L2Res | It provides access to a L2 resource. | symbloTeAPI | Any other SHAPES core component |

3.2.1.3 Test cases and Validation

Table 5 - **Interface symbloTeAPI-1.** This interface is used to register an existing ASAPA user to symbloTe.

| I/F | Test case | Method | Call | Result |
|---------------|-----------|--------|-------------------------------------------------------|-----------|
| symbloTeAPI-1 | sAPI-1.A | POST | <sAPI>:8082/symbiote/admin/registerUser/me/toSymbiote | Pass/Fail |

sAPI-1.A. This test case checks if a valid ASAPA user is registered to symbloTe successfully. Parameters required:

```
{
  "asapatoken": "string"
}
```

Response is given in JSON:

```
{
  "id": "string",
  "name": "string"
}
```

Table 6 - **Interface symbloTeAPI-2.** This interface is used to register a new L1 resource to symbloTe.

| I/F | Test case | Method | Call | Result |
|---------------|-----------|--------|----------------------------------------------|-----------|
| symbloTeAPI-2 | sAPI-2.A | POST | <sAPI>:8082/symbiote/resource/register/L1Res | Pass/Fail |

sAPI-2.A. This test case checks if an L1 resource is successfully registered to symbloTe.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

Parameters required:

```
{
  "platformusertoken": "string",
  "resourceinternalid": "string",
  "resourceplatformid": "string",
  "resourceinfo":{ List }
}
```

Response is given in JSON:

```
{
  "internalid": "string",
  "name": "string",
  "code": "string"
}
```

Table 7- **Interface symbloTeAPI-3**. This interface is used to get a list of L1 resources.

| I/F | Test case | Method | Call | Result |
|----------------------|-----------|--------|------------------------------------------|-----------|
| symbloTeAPI-3 | sAPI-3.A | POST | <sAPI>:8082 /symbiote/resource/get/L1Res | Pass/Fail |

sAPI-3.A. This test case checks if a list of L1 resources is returned.

Parameters required:

```
{
  "platformusertoken": "string",
  "resourceplatformid": "string"
}
```

Response is given in JSON:

```
List[{
  "name": "string",
  "id": "string"
}]
```

Table 8 - **Interface symbloTeAPI-4**. This interface is used to get resource information from an L1 resource using its name.

| I/F | Test case | Method | Call | Result |
|----------------------|-----------|--------|------------------------------------------|-----------|
| symbloTeAPI-4 | sAPI-4.A | POST | <sAPI>:8082 /symbiote/resource/get/L1Res | Pass/Fail |

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

sAPI-4.A. This test case checks if info about an L1 resource is returned.

Parameters required:

```
{
  "platformusertoken": "string",
  "resourceplatformid": "string",
  "resourcename": "string"
}
```

Response is given in JSON:

```
{
  "name": "string",
  "id": "string",
  "url": "string",
  "code": "string"
}
```

Table 9 - **Interface symbloTeAPI-5.** This interface is used to get resource information from an L1 resource using its id.

| I/F | Test case | Method | Call | Result |
|----------------------|-----------|--------|------------------------------------------|-----------|
| symbloTeAPI-5 | sAPI-5.A | POST | <sAPI>:8082 /symbiote/resource/get/L1Res | Pass/Fail |

sAPI-5.A. This test case checks if info about an L1 resource is returned.

Parameters required:

```
{
  "platformusertoken": "string",
  "resourceplatformid": "string",
  "resourceid": "string"
}
```

Response is given in JSON:

```
{
  "name": "string",
  "id": "string",
  "url": "string",
  "code": "string"
}
```


Table 10 - **Interface symbloTeAPI-6**. This interface is used to access an L1 resource using its name.

| I/F | Test case | Method | Call | Result |
|----------------------|-----------|--------|----------------------------------------------------|-----------|
| symbloTeAPI-6 | sAPI-6.A | POST | <sAPI>:8082 /symbiote/resource/access/sensor/L1Res | Pass/Fail |

sAPI-6.A. This test case checks if L1 resource can be accessed using its name.

Parameters required:

```
{
  "platformusertoken": "string",
  "resourceplatformid": "string",
  "resourcename": "string"
}
```

Response is given in JSON:

```
{
  "status": "string",
  "code": "string"
}
```

Table 11 - **Interface symbloTeAPI-7**. This interface is used to delete an L1 resource.

| I/F | Test case | Method | Call | Result |
|----------------------|-----------|--------|----------------------------------------------------|-----------|
| symbloTeAPI-7 | sAPI-7.A | POST | <sAPI>:8082 /symbiote/resource/access/sensor/L1Res | Pass/Fail |

sAPI-7.A. This test case checks if an L1 resource is successfully deleted.

Parameters required:

```
{
  "resourceplatformid": "string",
  "resourceinternalid": "string"
}
```

Response is given in JSON:

```
{
  "result": "string",
  "code": "string"
}
```

Table 12 - **Interface symbloTeAPI-8**. This interface is used to retrieve a list of L2 resources.

| I/F | Test case | Method | Call | Result |
|----------------------|-----------|--------|---------------------------------------------|-----------|
| symbloTeAPI-8 | sAPI-8.A | POST | <sAPI>:8082 /symbiote/resource/get/ListOfL2 | Pass/Fail |

sAPI-8.A. This test case checks if a list of L2 resources is successfully returned.

Parameters required:

```
{
  "paamusertoken": "string",
  "resourceplatformid": "string",
  "federationid": "string"
}
```

Response is given in JSON:

```
{
  "code": "string",
  "descriptpion": "string"
}
```

Table 13 - **Interface symbloTeAPI-9**. This interface is used to get information about an L2 resource.

| I/F | Test case | Method | Call | Result |
|----------------------|-----------|--------|------------------------------------------|-----------|
| symbloTeAPI-9 | sAPI-9.A | POST | <sAPI>:8082 /symbiote/resource/get/L2Res | Pass/Fail |

sAPI-9.A. This test case checks if information about L2 resources are successfully returned.

Parameters required:

```
{
  "paamusertoken": "string",
  "resourceplatformid": "string",
  "resourcename": "string",
  "resourceid": "string",
  "federationid": "string"
}
```

Response is given in JSON:

```
{
  "code": "string",
  "descriptpion": "string"
}
```

Table 14 - **Interface symbloTeAPI-10**. This interface is used to access an L2 resource.

| I/F | Test case | Method | Call | Result |
|-----------------------|-----------|--------|---------------------------------------------|-----------|
| symbloTeAPI-10 | sAPI-10.A | POST | <sAPI>:8082 /symbiote/resource/access/L2Res | Pass/Fail |

sAPI-10.A. This test case checks if an L2 resource can be successfully accessed.

Parameters required:

```
{
  "paamusertoken": "string",
  "resourceplatformid": "string",
  "resourcename": "string",
  "resourceid": "string",
  "federationid": "string"
}
```

Response is given in JSON:

```
{
  "code": "string",
  "descriptpion": "string"
}
```

3.2.2 Gateway (FINT)

SHAPES Gateway (GW) facilitates the interconnection of the edge IoT devices with the SHAPES Core cloud platform enabling as such the accommodation of the IoT collected data to the FInoT IoT platform (part of the SHAPES core); for more details, please see [1], section 6.2.2.

3.2.2.1 Overview of deployment options

SHAPES GW is to be deployed on pilot premises where the IoT edge sensors will reside. In this way, these sensors will be able to connect to the GW and forward their data where the existing FInoT middleware can be utilised for forwarding the gathered data to the FInoT platform. Prerequisites for the deployment are the existence of a power outlet (220V AC), and internet connectivity (i.e., the possibility to connect to the pilot's LAN or WLAN).

3.2.2.2 Interfaces offered

As documented in [1], section 6.2.2, the majority of the interfaces implemented from the SHAPES GW are internal and there is no possibility to access them directly from external entities (including the SHAPES core platform). In this context, the interfaces offered so far for testing external integration is for querying the GW status:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

Table 15 – SHAPES GW Interfaces.

| I/F | Feature | Endpoint/Queue | Description | Producer/Resource | Consumer/ Caller |
|------|----------|-----------------|----------------------------|-------------------|---------------------------------|
| REST | GET–live | <u>/gw/live</u> | Checks if the GW is online | GWAPI | Any other SHAPES core component |

3.2.2.3 Test cases and Validation

Table 16 - Interface GWAPI. This interface is used to check the GW status.

| I/F | Test case | Method | Call | Result |
|-------|-----------|--------|----------------------|-----------|
| GWAPI | gwAPI-1 | GET | <gwAPI>:9999/gw/live | Pass/Fail |

gwAPI-1. This test case checks if the GW is live. Parameters required:

Response is given in JSON:

```
{
  "id": "string",
  "status": "string"
}
```

3.2.3 FInoT IoT platform (FINT)

FInoT is a FIWARE-based IoT cloud management platform able to orchestrate and interconnect almost any kind of sensor, actuator and data logger (for more details please see [1], section 6.2.3.).

3.2.3.1 Overview of deployment options

FInoT is a 24/7 cloud based IoT platform, and it is not to be redeployed in the project's context. As such it will be available remotely, via a project wide available REST API (the API is included in [1], Annex 2, detailing preliminary API used by FInoT platform for communication with other core components”), using HTTPS as the secure communication protocol. Finally, there is no need for any specialised HW or SW for accessing the FInoT services; any standard browser or programming framework supporting REST API development (almost all modern programming frameworks do so) suffice for that purpose; the only requirement is to have a valid set of user credentials (created upon request).

3.2.3.2 Interfaces offered

The following interfaces are expected to be more in use for the project's needs:

Table 17- FINoT's platform interfaces

| I/F | Feature | Endpoint/Queue | Description | Producer / Resource | Consumer/ Caller |
|------|----------------------------------------|--------------------------------------|--------------------------------------------------------------------------------------------------------|---------------------|---------------------------------|
| REST | POST – Authenticate user | / auth/jwt/login | It authenticates an existing FINoT user to FINoT. | FINoT API | Any other SHAPES core component |
| REST | POST – Refresh the JWT token | auth/jwt/refresh | It refreshes the JWT token extending the 60 minutes, the user's valid session with the FINoT platform. | FINoT API | Any other SHAPES core component |
| REST | GET – List FINoT objects | inventory/v1/objects | Retrieves a list of objects which match the given criteria. | FINoT API | Any other SHAPES core component |
| REST | GET – Get FINoT object | inventory/v1/objects/<objectid> | Retrieves an object which match the given id. | FINoT API | Any other SHAPES core component |
| REST | GET – Get FINoT object historical data | inventory/v1/objects/<objectid>/data | Retrieves an object's historical data. | FINoT API | Any other SHAPES core component |

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

| | | | | | |
|------|------------------------------------|---------------------------------|----------------------------------------------------------------------------------------|-----------|---------------------------------|
| REST | POST – Create FInoT object | inventory/v1/objects | Creates an object (e.g., IoT measurement) in the FInoT platform | FInoT API | Any other SHAPES core component |
| REST | PATCH – Update FInoT object | inventory/v1/objects/<objectid> | Updates the object (e.g., IoT measurement) matching the given id in the FInoT platform | FInoT API | Any other SHAPES core component |
| REST | DELETE – Delete FInoT object | inventory/v1/objects/<objectid> | Deletes an object (e.g., IoT measurement) in the FInoT platform | FInoT API | Any other SHAPES core component |

For the complete offered interfaces, please refer to Annex 2 “FInoT Middleware API for SHAPES, detailing preliminary API used by FInoT platform for communication with other core components” of [1] deliverable.

3.2.3.3 Test cases and Validation

This section presents the test cases for the interfaces presented in the previous section.

Table 18 - **Interface finotAPI**. This interface is used to authenticate an existing user to FInoT.

| I/F | Test case | Method | Call | Result |
|----------|-----------|--------|-----------------------------|-----------|
| finotAPI | fAPI-1 | POST | <finotAPI>: /auth/jwt/login | Pass/Fail |

fAPI-1. This test case checks if a valid user is authenticating to FInoT successfully. Parameters required:

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

| Name | Type | Description | Required |
|----------|--------|--------------------|----------|
| loginId | String | User email address | Yes |
| password | String | User password | Yes |

| REQUEST BODY |
|-----------------------------------------------------------------------------|
| <pre>{ "loginId": "demouser@shapes.com", "password": "demodemo" }</pre> |

Response is given in JSON:

| Status | Response |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 200 | <pre>{ "status": 200, "result": { "token": "5cCI6lkpXVCIsCI6ljYzM...", "user": { "email": "demouser@shapes.com", "tenant": "shapes", "id": "ce363781-7ba0-4772-a0d8-04d231a2945f", "roles": ["role1", "role2"] }, "refreshToken": "IBnmvKadPxxkletcJzkpg1ctw..." } }</pre> |

3.2.3.3.1 Update JWT token test

Table 19 - Interface *finotAPI*. This interface is used to refresh the JWT authentication token.

| I/F | Test case | Method | Call | Result |
|----------|-----------|--------|-------------------------------|-----------|
| finotAPI | fAPI-2 | POST | <finotAPI>: /auth/jwt/refresh | Pass/Fail |

fAPI-2. This test case checks if a valid user presents a valid refresh token to the FINoT successfully. Parameters required:

| Name | Type | Description | Required |
|--------------|--------|---------------|----------|
| refreshToken | String | Refresh token | Yes |

| REQUEST BODY |
|---------------------------------------------------------------|
| <pre>{ "refreshToken": "ZfN7kxT9HwdkV_BIJQyYOaA..." }</pre> |

Response is given in JSON:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

| Status | Response |
|--------|-------------------------------------------------------------------------------------------|
| 200 | <pre>{ "status": 200, "result": { "token": "eyJhbGciOiJIUzI1NiUxQ..." } }</pre> |

3.2.3.3.2 List FiNoT objects test

Table 20 - **Interface finotAPI**. This interface is used to list all of SHAPES objects (i.e., IoT data) hosted in FiNoT.

| I/F | Test case | Method | Call | Result |
|----------|-----------|--------|----------------------------------|-----------|
| finotAPI | fAPI-3 | GET | <finotAPI>: inventory/v1/objects | Pass/Fail |

fAPI-3. This test case checks if a list of objects is returned from FiNoT successfully. Parameters required:

| Name | Type | Description | Required |
|---------------|--------|-------------------------------|----------|
| X-Tenant | String | Tenant name " shapes " | Yes |
| Authorization | String | JWT <Token> | Yes |

| |
|----------------------------------------------------------------------------------------|
| REQUEST HEADERS |
| X-Tenant: "shapes" Authorization: "JWT eyJhbGciOiJIUzI1NiUxQ...1NiIsInR5cCI6IkpXVC" |

Response is given in JSON:

| Status | Response |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 200 | <pre> { "status": 200, "result": { "objects": [{ "id": "ce331003-0069-48b9-a784-d09222a842ee", "type": "MyShapesObject", "Timestamp": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z", "metadata": {} }, "temperature": { "type": "Number", "value": "20", "metadata": { "Timestamp": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z" } } }, "location": { "type": "geo:json", "value": { "type": "Point", "coordinates": [45.466377, 13.654166] }, "metadata": {} } }] } } </pre> |

3.2.3.3.3 Get FiNoT object test

Table 21 - **Interface finotAPI**. This interface is used to get a specific SHAPES object (i.e., IoT data) hosted in FiNoT.

| I/F | Test case | Method | Call | Result |
|----------|-----------|--------|---------------------------------------------|-----------|
| finotAPI | fAPI-4 | GET | <finotAPI>: inventory/v1/objects/<objectId> | Pass/Fail |

fAPI-4. This test case checks if the objects specified by the given id is returned from FiNoT successfully. Parameters required:

| Name | Type | Description | Required |
|---------------|--------|-------------------------------|----------|
| X-Tenant | String | Tenant name " shapes " | Yes |
| Authorization | String | JWT <Token> | Yes |

REQUEST HEADERS

X-Tenant: "shapes"
Authorization: "JWT eyJhbGciOiJIUzI...1NiIsInR5cCI6IkpXVC"

Response is given in JSON:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

| Status | Response |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 201 | <pre>{ "status": 201, "result": { "object": { "id": "ce331003-0069-48b9-a784-d09222a842ee", "type": "MyShapesObject", "TimelInstant": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z", "metadata": {} }, "temperature": { "type": "Number", "value": "20", "metadata": { "TimelInstant": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z" } } }, "location": { "type": "geo:json", "value": { "type": "Point", "coordinates": [45.466377, 13.654166] } }, "metadata": {} } } }</pre> |

3.2.3.3.4 Get FiNoT object historical data test

Table 22 - **Interface finotAPI**. This interface is used to get a specific SHAPES object (i.e., IoT data) historical data hosted in FiNoT.

| I/F | Test case | Method | Call | Result |
|----------|-----------|--------|--------------------------------------------------|-----------|
| finotAPI | fAPI-5 | GET | <finotAPI>: inventory/v1/objects/<objectId>/data | Pass/Fail |

fAPI-5. This test case checks if the specified by the given id object's historical data is returned from FiNoT successfully.

Parameters required:

| Name | Type | Description | Required |
|---------------|--------|-------------------------------|----------|
| X-Tenant | String | Tenant name " shapes " | Yes |
| Authorization | String | JWT <Token> | Yes |

REQUEST HEADERS

X-Tenant: "shapes"
Authorization: "JWT eyJhbGciOiJIUzIuLCI6IkpXVC"

Response is given in JSON:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

| Status | Response |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 200 | <pre> { "status": 200, "result": { "data": { "attributes": [{ "attrName": "temperature", "values": [12.612307695242075, 11.03249986966451, 11.437272678722035, 8.432499885559082] }], "entityId": "ce331003-0069-48b9-a784-d09222a842ee", "index": ["2019-12-03T00:00:00.000", "2019-12-04T00:00:00.000", "2019-12-05T00:00:00.000", "2019-12-08T00:00:00.000"] } } } </pre> |

3.2.3.3.5 Create FiNoT object test

Table 23 - Interface *finotAPI*. This interface is used to create a SHAPES object (i.e., IoT data) in FInoT.

| I/F | Test case | Method | Call | Result |
|----------|-----------|--------|---------------------------------------------|-----------|
| finotAPI | fAPI-6 | POST | <finotAPI>: inventory/v1/objects/<objectId> | Pass/Fail |

fAPI-6. This test case checks if an object is created in FInoT successfully. Parameters required:

| Name | Type | Description | Required |
|---------------|--------|-------------------------------|----------|
| X-Tenant | String | Tenant name " shapes " | Yes |
| Authorization | String | JWT <Token> | Yes |

| REQUEST BODY |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> { "type": "MyShapesObject", "attributes": [{ "name": "temperature", "type": "Number", "value": "20" }], "static_attributes": [{ "name": "location", "type": "geo:json", "value": { "type": "Point", "coordinates": [37.996523, 23.814874] } }] } </pre> |

Response is given in JSON:

| Status | Response |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 201 | <pre> { "status": 201, "result": { "object": { "id": "ce331003-0069-48b9-a784-d09222a842ee", "type": "MyShapesObject", "Timestamp": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z", "metadata": {} }, "temperature": { "type": "Number", "value": "20", "metadata": { "Timestamp": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z" } } }, "location": { "type": "geo:json", "value": { "type": "Point", "coordinates": [45.466377, 13.654166] }, "metadata": {} } } } } </pre> |

3.2.3.3.6 Update FiNoT object test

Table 24- **Interface finotAPI**. This interface is used to update a SHAPES object (i.e., IoT data) in FiNoT.

| I/F | Test case | Method | Call | Result |
|-----|-----------|--------|------|--------|
|-----|-----------|--------|------|--------|

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

| | | | | |
|----------|--------|-------|---------------------------------------------|-----------|
| finotAPI | fAPI-7 | PATCH | <finotAPI>: inventory/v1/objects/<objectId> | Pass/Fail |
|----------|--------|-------|---------------------------------------------|-----------|

fAPI-7. This test case checks if an object is updated in FInoT successfully. Parameters required:

| Name | Type | Description | Required |
|---------------|--------|-------------------------------|----------|
| X-Tenant | String | Tenant name " shapes " | Yes |
| Authorization | String | JWT <Token> | Yes |

| REQUEST HEADERS |
|----------------------------------------------------------------------------------|
| X-Tenant: "shapes" Authorization: "JWT eyJhbGciOiJIUzI...1NilsInR5cCI6IkpXVC" |
| REQUEST BODY |
| { "temperature": { "type": "Number", "value": 20.5 } } |

Response is given in JSON:

| Status | Response |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 200 | { "status": 200, "result": { "object": { "id": "ce331003-0069-48b9-a784-d09222a842ee", "type": "MyShapesObject", "Timestamp": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z", "metadata": {} }, "temperature": { "type": "Number", "value": "30", "metadata": { "Timestamp": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z" } } }, "location": { "type": "geo:json", "value": { "type": "Point", "coordinates": [45.466377, 13.654166] } }, "metadata": {} } } } |

3.2.3.3.7 Delete FiNoT object test

Table 25 - **Interface finotAPI**. This interface is used to delete a SHAPES object (i.e., IoT data, current and historical) in FInoT.

| I/F | Test case | Method | Call | Result |
|----------|-----------|--------|---------------------------------------------|-----------|
| finotAPI | fAPI-8 | DELETE | <finotAPI>: inventory/v1/objects/<objectId> | Pass/Fail |

fAPI-8. This test case checks if an object is deleted in FInoT successfully. Parameters required:

| Name | Type | Description | Required |
|---------------|--------|----------------------|----------|
| X-Tenant | String | Tenant name "shapes" | Yes |
| Authorization | String | JWT <Token> | Yes |

REQUEST HEADERS

X-Tenant: "shapes"
Authorization: "JWT eyJhbGciOiJIUzIuLnR5cCI6IkpXVC"

Response:

| Status | Response |
|--------|----------|
| 204 | |

3.2.4 FHIR Medical Interoperability (GNO)

The FHIR medical interoperability component facilitates the interoperability and communication among digital solutions that exchange medical-related information with each other and/or other SHAPES core components. The main component of the FHIR interoperability is the Message Queue (MQ).

3.2.4.1 Overview of deployment options

The FHIR MQ is a simple REST API component based on Spring technology. It offers a simple endpoint secured by Request Header API Key, and is used for exchanging data between Digital Solutions. It is integrated with Apache Kafka. The FHIR MQ is deployed using Docker technology and is a set of 3 Docker containers. It can work with any Linux Distribution but is tested under Ubuntu 18+.

3.2.4.2 Interfaces offered

Table 26 – GNOMON's interfaces.

| I/F | Feature | Endpoint/Queue | Description | Producer/Resource | Consumer/ Caller |
|------|------------------------------------|-------------------------------------|------------------------------|-------------------|---------------------------------|
| REST | POST –Exchange Data from Solutions | /MeasurementService/rest/mq/publish | Exchange Data from Solutions | FHIR MQ | Any other SHAPES core component |

3.2.4.3 Test cases and Validation

Table 27 – FHIR MQ Test Cases

| I/F | Test Case | Description | Result |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 1 | <p>DS receives FHIR data from FHIR MQ</p> <p>Prerequisite</p> <p>DS has registered to FHIR MQ by sending Company name, IP Address and endpoint for sending the messages to its solution</p> | <p>1. DS sends FHIR data to FHIR MQ</p> <p>2. DS receive FHIR data from FHIR MQ</p> | <p>1. FHIR should respond with 200 OK</p> <p>2. Other DS's who are registered can receive in their registered endpoint the payload.</p> |
| 2 | <p>DS sends data to FHIR MQ</p> <p>Prerequisite</p> <p>DS must have an API-KEY</p> | <p>1. Send FHIR payload to endpoint with correct API-key</p> <p>2. Send FHIR payload to endpoint with wrong API-key</p> <p>3. Send FHIR payload to endpoint with non-validated payload</p> | <p>1. REST API responds with 200 OK</p> <p>2. REST API responds with 403 Forbidden</p> <p>3. REST API responds with 400 Bad Request</p> |

3.2.5 Big Data Platform: Data Lakehouse & Analytics Engine (TREE)

The Big Data Platform combines the Data Lakehouse with the Analytics Engine, allowing Digital Solutions to send their data to the Data Lakehouse for advance processing using the AI-based Analytics Engine.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

3.2.5.1 *Overview of deployment options*

The technologies used for the implementation of the different components of the Big Data Platform component are open-source, referential and state-of-the-art technologies in their respective topics. In this way, the entire implementation of the Platform is decoupled from any specific cloud services.

The Platform is deployed in AWS in three different stages: a first phase that takes advantage of the required technologies, exposed as AWS services (i.e. BDaaS), a second phase that uses the platforms managed by AWS that provide the required technologies (i.e. PaaS) and a third phase in which only the infrastructure offered by amazon is used (i.e. IaaS) to deploy on it, the required platforms.

In this way, the setup of the Development environment is initially facilitated, enabling progress in the implementation of the Project and, during the progression of the Project, it is gradually decoupled from the management services offered by AWS.

The inbound and outbound API interfaces for the Big Data Platform have been implemented in Python Flask and in Node.js + Amazon Athena technologies. The inbound API should be configured at launch time with the path to the Tier-0 (RAW data staging) bucket of the Amazon S3. For the outbound API, the SQL tables should be created in Amazon Athena corresponding to the formats of the Delta data files stored in the Amazon S3 Tier-3 (analysis results) bucket. For example, for sleep activity this SQL create query has been used to create SQL table in Amazon Athena:

```
CREATE EXTERNAL TABLE `api2_out_sleep_act` (  
  `user_id` string,  
  `date` string,  
  `start_time` timestamp,  
  `latency` double,  
  `in_bed` double,  
  `sleep_duracion` double,  
  `sleep_efficiency` double,  
  `n_int_away` int,
```

```
  `n_int_getup` int,  
  `int_duration` double,  
  `sqi` double,  
  `sleep_disconnect` double) ROW FORMAT SERDE  
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'  
  STORED AS INPUTFORMAT  
  'org.apache.hadoop.mapred.TextInputFormat' OUTPUTFORMAT  
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'  
  LOCATION 's3://h2020-shapes-dev/datalakehouse/02-enriched-  
tier/api2_out.sleep_activity.delta'  
  TBLPROPERTIES ( 'has_encrypted_data'='false',  
    'transient_lastDdlTime'='1626440336' )
```

Physical activity SQL table can be created in Amazon Athena with the following SQL command:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.


```
CREATE EXTERNAL TABLE `api2_out_phys_act` (
  `user_id` string,
  `date` date,
  `steps` int,
  `sedentary` int,
  `active` int,
  `adl` int,
  `intermediate` int,
  `exercise` int,
  `light_exercise` int,
  `moderate_exercise` int,
  `intense_exercise` int,
  `rest_exercise` int,
  `disconnect` int) ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
  STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
  OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
  LOCATION
  's3://h2020-shapes-dev/datalakehouse/02-enriched-
  tier/api2_out.phys_activity.delta'
  TBLPROPERTIES ( 'has_encrypted_data'='false',
    'transient_lastDdlTime'='1626441698' )
```

For deployment, two Docker images have been built, one for the inbound and another for the outbound APIs. For the deployment of these Docker containers into a production environment, those containers should have configured with access to the Big Data Platform S3 Amazon storage (either to be on the same Amazon cluster, or to be within the IP ranges allowed by the Amazon Cloud network / firewall access to the corresponding S3 Delta Lake cluster).

3.2.5.2 *Interfaces offered*

At a global scale, the Big Data Platform has two major interfaces with the other systems, both with the HTTP REST protocol:

- The inbound data API which allows the raw IoT and FHIR data to be imported from FINoT (through the symbloTe connector) and the FHIR connector into the SHAPES Big Data Platform, respectively. In order to scale up the number of data records passed per HTTP API call, the data is transferred in a form of

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

JSON file payloads, which are stored into Tier-0 (RAW data) staging of the Platform on Amazon S3 for further data processing (cleansing, ETL, analysis, etc). This API interface has been implemented in Python Flask.

- The outbound data API which allows the results of the data analysis carried out in the Big Data Platform to be fetched from the Platform into the Digital Solutions system. These data are composed of the initial log-like IoT sensor data plus a reconstructed data per each original data record, thus, there is no row_id or object_id and API calls fetch data filtered either per user_id value or per dates values. This API has been implemented with Node.js reading data from the Amazon S3 Tier-3 (results data) storage of the Platform via Amazon Athena SQL-like service.

For both the inbound and outbound API interfaces, the authentication has been implemented based on JWT token technology. New user should sign up once to set up his/her username and password. Once signed up, he/she should sign in for every working session, providing username/password and receiving back an authorization JWT token, which should be used in all other regular (get/post) API HTTP calls.

The API call specifications are presented below per inbound and outbound interfaces respectively.

Table 28 - Big Data Platform inbound API:

| I/F | Feature | Endpoint/Queue | Description | Producer/Resource | Consumer/Caller |
|------|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-----------------------|-------------------|
| REST | POST – register user | PATH: /v1/signup HTTP header: 'content-type: application/json' HTTP data: '{"username": "my_user", "password": "my_pas123"}' | Register a new user | Big Data Platform API | Digital Solutions |
| REST | POST – login existing user | PATH: /v1/signin HTTP header: | Log in existing user | Big Data Platform API | Digital Solutions |

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

| I/F | Feature | Endpoint/Queue | Description | Producer/ Resource | Consumer/ Caller |
|------|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|--------------------------|---------------------|
| | | 'content-type: application/json' HTTP data: '{"username": "my_user", "password": "my_pas123"}' | | | |
| REST | POST – upload JSON data | PATH: /v1/post_data HTTP header: "x-access-token: \${TOKEN}" "Content-Type: application/json" HTTP data: <FHIR json data> | Uploads JSON data to Tier-0 AWS S3 storage | Big Data Platform API | FHIR |

Table 29 - Big Data Platform outbound API:

| I/F | Feature | Endpoint/Queue | Description | Producer / Resource | Consumer / Caller |
|------|---------------------------------------|------------------------------------------------------------------------------------|--------------------------------------------------------------------|-----------------------------|-------------------------|
| REST | GET – fetch sleep activity data | PATH: /v1/sleep_act HEADER: "x-access-token: \${TOKEN}" | Get the N latest sleep activity data rows | Big Data Platform API | Digital Solutions |
| REST | GET – fetch sleep activity data | PATH: /v1/sleep_act?user_id=00001 HEADER: "x-access-token: \${TOKEN}" | Get sleep activity data rows for a given user_id value | Big Data Platform API | Digital Solutions |

| I/F | Feature | Endpoint/Queue | Description | Producer / Resource | Consumer / Caller |
|------|------------------------------------|------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|-----------------------|-------------------|
| REST | GET – fetch sleep activity data | PATH: /v1/sleep_act?start_date=2019-10-27&end_date=2019-10-27 HEADER: "x-access-token: \${TOKEN}" | Get sleep activity data rows for a given date range | Big Data Platform API | Digital Solutions |
| REST | GET – fetch sleep activity data | PATH: /v1/sleep_act?user_id=00001&start_date=2019-10-27&end_date=2019-10-27 HEADER: "x-access-token: \${TOKEN}" | Get sleep activity data rows for a given user_id value and given date range | Big Data Platform API | Digital Solutions |
| REST | GET – fetch physical activity data | PATH: /v1/phys_act HEADER: "x-access-token: \${TOKEN}" | Get the N latest physical activity data rows | Big Data Platform API | Digital Solutions |
| REST | GET – fetch physical activity data | PATH: /v1/phys_act?user_id=00001 HEADER: "x-access-token: \${TOKEN}" | Get physical activity data rows for a given user_id value | Big Data Platform API | Digital Solutions |
| REST | GET – fetch physical activity data | PATH: /v1/phys_act?start_date=2019-10-27&end_date=2019-10-27 HEADER: "x-access-token: \${TOKEN}" | Get physical activity data rows for a given date range | Big Data Platform API | Digital Solutions |
| REST | GET – fetch physical activity data | PATH: /v1/phys_act?user_id=00001&start_date=2019-10-27&end_date=2019-10-27 | Get physical activity data rows for a | Big Data Platform API | Digital Solutions |

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

| I/F | Feature | Endpoint/Queue | Description | Producer / Resource | Consumer / Caller |
|------|------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|-----------------------|-------------------|
| | | HEADER: "x-access-token: \${TOKEN}" | given user_id value and given a date range | | |
| REST | GET – fetch vitals control data | PATH: /v1/vitals_control HEADER: "x-access-token: \${TOKEN}" | Get the N latest vitals control data rows | Big Data Platform API | Digital Solutions |
| REST | GET – fetch vitals control data | PATH: /v1/vitals_control?user_id=0001 HEADER: "x-access-token: \${TOKEN}" | Get vitals control data rows for a given user_id value | Big Data Platform API | Digital Solutions |
| REST | GET – fetch vitals control data | PATH: /v1/vitals_control?start_date=2021-06-01&end_date=2021-06-01 HEADER: "x-access-token: \${TOKEN}" | Get vitals control data rows for a given date range | Big Data Platform API | Digital Solutions |
| REST | GET – fetch vitals control data | PATH: /v1/vitals_control?user_id=0001&start_date=2021-06-01&end_date=2021-06-01 HEADER: "x-access-token: \${TOKEN}" | Get vitals control data rows for a given user_id value and given a date range | Big Data Platform API | Digital Solutions |
| REST | GET – fetch anomaly detection data | PATH: /v1/anomaly_detect HEADER: "x-access-token: \${TOKEN}" | Get the N latest anomaly detection data rows | Big Data Platform API | Digital Solutions |

| I/F | Feature | Endpoint/Queue | Description | Producer / Resource | Consumer / Caller |
|------|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|-----------------------|-------------------|
| REST | GET – fetch anomaly detection data | PATH: /v1/anomaly_detect?user_id=00001 HEADER: "x-access-token: \${TOKEN}" | Get anomaly detection data rows for a given user_id value | Big Data Platform API | Digital Solutions |
| REST | GET – fetch anomaly detection data | PATH: /v1/anomaly_detect?start_date=2019-11-03&end_date=2019-11-03 HEADER: "x-access-token: \${TOKEN}" | Get anomaly detection data rows for a given date range | Big Data Platform API | Digital Solutions |
| REST | GET – fetch anomaly detection data | PATH: /v1/anomaly_detect?user_id=00001&start_date=2019-11-03&end_date=2019-11-03 HEADER: "x-access-token: \${TOKEN}" | Get anomaly detection data rows for a given user_id value and given a date range | Big Data Platform API | Digital Solutions |

3.2.5.3 Test cases and Validation

The Big Data Platform **inbound API** is used to send JSON files with IoT sensor data to the SHAPES Big Data Platform.

Table 30 - **Case-INB-01**. This test case checks if a new user can be registered to the inbound API successfully.

| I/F | Test case | Method | Call | Result |
|---------------------------|-----------|--------|---------------------------------------------------------------------------------------------------------------------------------|-----------|
| Big Data Platform Inbound | INB-01 | POST | curl -X POST --header 'content-type: application/json' -d '{"username": "user123", "password": "3"}' \$BASE_URL/v1/signup | Pass/Fail |

Test parameters:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

```
{
  "username": "new_user123",
  "password": "new_pass123"
}
```

Response JSON in case of test success:

```
{
  "ok": 1,
  "new_id": "new_user_id_XYZ"
}
```

Table 31 - **Case-INB-02**. This test case checks if a new user can be logged in to the inbound API successfully.

| I/F | Test case | Method | Call | Result |
|---------------------------|-----------|--------|---------------------------------------------------------------------------------------------------------------------------|-----------|
| Big Data Platform Inbound | INB-02 | POST | curl -X POST --header 'content-type: application/json' -d '{"username": "user123", "password": "3"}' \$BASE_URL/v1/signin | Pass/Fail |

Test parameters (the same as in Case-INB-01):

```
{
  "username": "new_user123",
  "password": "new_pass123"
}
```

Response JSON in case of test success:

```
{
  "ok": 1,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjV6M2JrIiwiaWF0IjoxNjMxNTM0MDYxLCJleHAiOiJlMzE2MjA0NjF9.AqO_PsFXTxElzxluFOK4EjVpVfhfFqtHwrk2yYEqluI"
}
```

Table 32 - **Case-INB-03**. This test case checks if a FHIR data JSON can be uploaded to the inbound API successfully.

| I/F | Test case | Method | Call | Result |
|---------------------------|-----------|--------|------------------------------------------------------------------------------------------------------------------------------|-----------|
| Big Data Platform Inbound | INB-03 | POST | curl -i -X POST -H "x-access-token: \$TOKEN" -H "Content-Type: application/json" -d <FHIR json data> \$BASE_URL/v1/post_data | Pass/Fail |

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

Test parameters are passed through HTTP Header & Form data fields (the TOKEN to be the same as returned in Case-INB-02):

```
-H "x-access-token: <$TOKEN>"
-H "Content-Type: application/json"
-d <FHIR json data>
```

Response JSON in case of test success:

```
{
  "ok": 1
}
```

As for the Big Data Platform **outbound API**, it is used to convey analysis data from the SHAPES Big Data Platform into Digital Solutions system.

Table 33 - **Case-OUT-01**. This test case checks if sleep activity data can be fetched from the outbound API successfully.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-01 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/sleep_act" | Pass/Fail |

Test parameter in this case is the authentication TOKEN passed in the HTTP header:

```
-H "x-access-token: ${TOKEN}"
```

Response JSON example in case of test success:

```
{
  "ok": 1,
  "res": { "Items": [ {"user_id": "00001",
    "date": "2019-10-27",
    "start_time": "2019-10-27 00:38:00.000",
    "latency":3,
    "in_bed":531,
    "sleep_efficiency":0.9698681732580039,
    "n_int_away":3,
    "n_int_getup":0,
    "int_duration":13,
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.


```

        "sqi":0.8999999999999999,
        "sleep_disconnect":0
    }
}
}

```

Table 34 - **Case-OUT-02.** This test case checks if sleep activity data can be fetched from the outbound API successfully for a given user_id.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|--------------------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-02 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/sleep_act?user_id=00077" | Pass/Fail |

Test parameters in this case are the authentication TOKEN passed in the HTTP header and a user_id value passed in the URL:

```

-H "x-access-token: ${TOKEN}"
?user_id=00077

```

Response JSON example in case of test success:

```

{
  "ok": 1,
  "res": { "Items": [ {"user_id": "00077",
    "date": "2019-12-07",
    "start_time": "2019-12-07 01:31:00.000",
    "latency":3,
    "in_bed":431,
    "sleep_efficiency":0.67986732580039,
    "n_int_awake":2,
    "n_int_getup":0,
    "int_duration":12,
    "sqi":0.733325329,
    "sleep_disconnect":0
  }
  ]
}
}

```

Table 35 - **Case-OUT-03**. This test case checks if sleep activity data can be fetched from the outbound API successfully for a given a date range.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|------------------------------------------------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-03 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/sleep_act?start_date=2019-10-28&end_date=2019-10-28" | Pass/Fail |

Test parameters in this case are authentication the TOKEN passed in the HTTP header and the date range values passed in the URL:

```
-H "x-access-token: ${TOKEN}"
?start_date=2019-10-28&end_date=2019-10-28
```

Response JSON example in case of test success:

```
{
  "ok": 1,
  "res": { "Items": [ { "user_id": "00001",
    "date": "2019-10-28",
    "start_time": "2019-10-28 00:17:00.000",
    "latency": 3,
    "in_bed": 641,
    "sleep_efficiency": 0.7986732580039,
    "n_int_awake": 3,
    "n_int_getup": 0,
    "int_duration": 15,
    "sqi": 0.911235329,
    "sleep_disconnect": 0
  }
  ]
}
```

Table 36 - **Case-OUT-04**. This test case checks if sleep activity data can be fetched from the outbound API successfully for a given user_id and a date range.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|--------------------------------------------------------------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-04 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/sleep_act?user_id=00077&start_date=2019-10-28&end_date=2019-10-28" | Pass/Fail |

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

Test parameters in this case are the authentication TOKEN passed in the HTTP header and the user_id and the date range values passed in the URL:

```
-H "x-access-token: ${TOKEN}"
?user_id=00077&start_date=2019-10-28&end_date=2019-10-28
```

Response JSON example in case of test success:

```
{
  "ok": 1,
  "res": { "Items": [ { "user_id": "00077",
    "date": "2019-10-28",
    "start_time": "2019-10-28 00:09:12.000",
    "latency":1,
    "in_bed":662,
    "sleep_efficiency":0.8621732580039,
    "n_int_awake":2,
    "n_int_getup":0,
    "int_duration":7,
    "sqi":0.9721135329,
    "sleep_disconnect":0
  }
  ]
}
}
```

Table 37 - **Case-OUT-05.** This test case checks if physical activity data can be fetched from the outbound API successfully.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|-----------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-05 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/phys_act" | Pass/Fail |

Test parameter in this case is the authentication TOKEN passed in the HTTP header:

```
-H "x-access-token: ${TOKEN}"
```

Response JSON example in case of test success:

```
{
  "ok": 1,
  "res": { "Items": [ { "user_id": "00001",
    "date": "2019-12-07",
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

```

        "steps":2839,
        "sedentary":813,
        "active":109,
        "adl":83,
        "intermediate":0,
        "exercise":26,
        "light_exercise":17,
        "moderate_exercise":8,
        "intense_exercise":0,
        "rest_exercise":1
    }
]
}

```

Table 38 - **Case-OUT-06**. This test case checks if physical activity data can be fetched from the outbound API successfully for a given user_id.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|-------------------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-06 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/phys_act?user_id=00077" | Pass/Fail |

Test parameters in this case are the authentication TOKEN passed in the HTTP header and a user_id value passed in the URL:

```

-H "x-access-token: ${TOKEN}"
?user_id=00077

```

Response JSON example in case of test success:

```

{
  "ok": 1,
  "res": { "Items": [ { "user_id": "00077",
    "date": "2019-12-07",
    "steps":1467,
    "sedentary":213,
    "active":124,
    "adl":88,
    "intermediate":0,
    "exercise":21,
    "light_exercise":19,
    "moderate_exercise":8,
    "intense_exercise":0,
    "rest_exercise":1
  }
  ]
}

```

```

    ]
  }
}

```

Table 39 - **Case-OUT-07**. This test case checks if physical activity data can be fetched from the outbound API successfully for given a date range.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|-----------------------------------------------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-07 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/phys_act?start_date=2019-10-28&end_date=2019-10-28" | Pass/Fail |

Test parameters in this case are the authentication TOKEN passed in the HTTP header and the date range values passed in the URL:

```

-H "x-access-token: ${TOKEN}"
?start_date=2019-10-28&end_date=2019-10-28

```

Response JSON example in case of test success:

```

{
  "ok": 1,
  "res": { "Items": [ {"user_id": "00001",
    "date": "2019-10-28",
    "steps":2386,
    "sedentary":193,
    "active":287,
    "adl":67,
    "intermediate":0,
    "exercise":34,
    "light_exercise":14,
    "moderate_exercise":8,
    "intense_exercise":0,
    "rest_exercise":1
  }
  ]
}

```

Table 40 - **Case-OUT-08.** This test case checks if physical activity data can be fetched from the outbound API successfully for a given user_id and a date range.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|-------------------------------------------------------------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-08 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/phys_act?user_id=00077&start_date=2019-10-28&end_date=2019-10-28" | Pass/Fail |

Test parameters in this case are the authentication TOKEN passed in the HTTP header and a user_id and the date range values passed in the URL:

```
-H "x-access-token: ${TOKEN}"  
?user_id=00077&start_date=2019-10-28&end_date=2019-10-28
```

Response JSON example in case of test success:

```
{  
  "ok": 1,  
  "res": { "Items": [ { "user_id": "00077",  
                        "date": "2019-10-28",  
                        "steps":2386,  
                        "sedentary":193,  
                        "active":287,  
                        "adl":67,  
                        "intermediate":0,  
                        "exercise":34,  
                        "light_exercise":14,  
                        "moderate_exercise":8,  
                        "intense_exercise":0,  
                        "rest_exercise":1  
                      }  
            ]  
        }  
}
```

Table 41 - **Case-OUT-09.** This test case checks if vitals control can be fetched from the outbound API successfully.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|-----------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-09 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/vitals_control" | Pass/Fail |

Test parameter in this case is the authentication TOKEN passed in the HTTP header:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

```
-H "x-access-token: ${TOKEN}"
```

Response JSON example in case of test success:

```
{
  "ok": 1,
  "res": { "Items": [ { "user_id": "00001",
    "timestamp": 1622531451000,
    "bg": 87,
    "wr_lcl": 68.24,
    "wr_ucl": 160.05,
    "lcl": 45.29,
    "ucl": 182.00,
    "status": "ic"
  },
    { "user_id": "00001",
    "timestamp": 1622538612000,
    "bg": 136,
    "wr_lcl": 68.24,
    "wr_ucl": 160.05,
    "lcl": 45.29,
    "ucl": 182.00,
    "status": "ic"
  }
  ]
}
```

Table 42 - **Case-OUT-10**. This test case checks if vitals control data can be fetched from the outbound API successfully for a given user_id.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|-------------------------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-10 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/vitals_control?user_id=00077" | Pass/Fail |

Test parameters in this case are the authentication TOKEN passed in the HTTP header and a user_id value passed in the URL:

```
-H "x-access-token: ${TOKEN}"
?user_id=00077
```

Response JSON example in case of test success:

```
{
  "ok": 1,
  "res": { "Items": [ { "user_id": "00077",
    "timestamp": 1622531451000,
    "bg": 87,
    "wr_lcl": 68.24,
    "wr_ucl": 160.05,
    "lcl": 45.29,
    "ucl": 182.00,
    "status": "ic"
  },
    { "user_id": "00077",
    "timestamp": 1622538612000,
    "bg": 136,
    "wr_lcl": 68.24,
    "wr_ucl": 160.05,
    "lcl": 45.29,
    "ucl": 182.00,
    "status": "ic"
  }
  ]
}
```

Table 43 - **Case-OUT-11**. This test case checks if vitals control can be fetched from the outbound API successfully for a given a date range.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|-----------------------------------------------------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-11 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/vitals_control?start_date=2021-06-01&end_date=2021-06-01" | Pass/Fail |

Test parameters in this case are the authentication TOKEN passed in the HTTP header and the date range values passed in the URL:

```
-H "x-access-token: ${TOKEN}"
?start_date=2021-06-01&end_date=2021-06-01
```

Response JSON example in case of test success:

```
{
  "ok": 1,
  "res": { "Items": [ { "user_id": "00001",
    "date": "2021-06-01",
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.


```

        "timestamp":1622531451000,
        "bg":87,
        "wr_lcl":68.24,
        "wr_ucl":160.05,
        "lcl":45.29,
        "ucl":182.00,
        "status":"ic"
      },
      {
        "user_id":"00001",
        "date":"2021-06-01",
        "timestamp":1622538612000,
        "bg":136,
        "wr_lcl":68.24,
        "wr_ucl":160.05,
        "lcl":45.29,
        "ucl":182.00,
        "status":"ic"
      }
    ]
  }
}

```

Table 44 - **Case-OUT-12**. This test case checks if vitals control data can be fetched from the outbound API successfully for a given user_id and a date range.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|-------------------------------------------------------------------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-12 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/vitals_control?user_id=00077&start_date=2021-06-01&end_date=2021-06-01" | Pass/Fail |

Test parameters in this case are the authentication TOKEN passed in the HTTP header and a user_id and the date range values passed in the URL:

```

-H "x-access-token: ${TOKEN}"
?user_id=00077&start_date=2021-06-01&end_date=2021-06-01

```

Response JSON example in case of test success:

```

{
  "ok": 1,
  "res": {
    "Items": [
      {
        "user_id": "00077",
        "date": "2021-06-01",
        "timestamp": 1622531451000,
        "bg": 87,
        "wr_lcl": 68.24,

```

```

        "wr_ucl":160.05,
        "lcl":45.29,
        "ucl":182.00,
        "status":"ic"
    },
    {
        "user_id":"00077",
        "date":"2021-06-01",
        "timestamp":1622538612000,
        "bg":136,
        "wr_lcl":68.24,
        "wr_ucl":160.05,
        "lcl":45.29,
        "ucl":182.00,
        "status":"ic"
    }
]
}

```

Table 45 - **Case-OUT-13**. This test case checks if anomaly detection data can be fetched from the outbound API successfully.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|-----------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-13 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/anomaly_detect" | Pass/Fail |

Test parameter in this case is the authentication TOKEN passed in the HTTP header:

```
-H "x-access-token: ${TOKEN}"
```

Response JSON example in case of test success:

```

{
  "ok": 1,
  "res": {
    "Items": [
      {
        "user_id": "00001",
        "init": 1572740580000,
        "end": 1572771060000,
        "activity": "sleep",
        "anomaly": "no_anomaly"
      },
      {
        "user_id": "00001",
        "init": 1572771060000,
        "end": 1572777600000,
        "activity": "kitchen",
        "anomaly": "no_anomaly"
      }
    ]
  }
}

```

```

    ]
  }
}

```

Table 46 - **Case-OUT-14** This test case checks if anomaly detection data can be fetched from the outbound API successfully for a given user_id.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|-------------------------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-14 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/anomaly_detect?user_id=00077" | Pass/Fail |

Test parameters in this case are the authentication TOKEN passed in the HTTP header and a user_id value passed in the URL:

```

-H "x-access-token: ${TOKEN}"
?user_id=00077

```

Response JSON example in case of test success:

```

{
  "ok": 1,
  "res": {"Items": [ {"user_id": "00077",
    "init": 1572740580000,
    "end": 1572771060000,
    "activity": "sleep",
    "anomaly": "no_anomaly"
  },
    {"user_id": "00077",
    "init": 1572771060000,
    "end": 1572777600000,
    "activity": "kitchen",
    "anomaly": "no_anomaly"
  }
  ]
}

```

Table 47 - **Case-OUT-15**. This test case checks if anomaly detection data can be fetched from the outbound API successfully for a given a date range.

| I/F | Test case | Method | Call | Result |
|----------------------------|-----------|--------|-----------------------------------------------------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-15 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/anomaly_detect?start_date=2019-11-03&end_date=2019-11-03" | Pass/Fail |

Test parameters in this case are the authentication TOKEN passed in the HTTP header and the date range values passed in the URL:

```
-H "x-access-token: ${TOKEN}"
?start_date=2019-11-03&end_date=2019-11-03
```

Response JSON example in case of test success:

```
{
  "ok": 1,
  "res": {"Items": [ {"user_id": "00001",
    "date": "2019-11-03",
    "init": 1572740580000,
    "end": 1572771060000,
    "activity": "sleep",
    "anomaly": "no_anomaly"
  },
    {"user_id": "00001",
    "date": "2019-11-03",
    "init": 1572771060000,
    "end": 1572777600000,
    "activity": "kitchen",
    "anomaly": "no_anomaly"
  }
  ]
}
```

Table 48 - **Case-OUT-16**. This test case checks if anomaly detection data can be fetched from the outbound API successfully for a given user_id and a date range.

| I/F | Test case | Method | Call | Result |
|-----|-----------|--------|------|--------|
|-----|-----------|--------|------|--------|

| | | | | |
|-----------------------------------|--------|-----|-------------------------------------------------------------------------------------------------------------------------------|-----------|
| Big Data Platform Outbound | OUT-16 | GET | curl -H "x-access-token: \${TOKEN}" "\$BASE_URL/v1/anomaly_detect?user_id=00077&start_date=2019-11-03&end_date=2019-11-03" | Pass/Fail |
|-----------------------------------|--------|-----|-------------------------------------------------------------------------------------------------------------------------------|-----------|

Test parameters in this case are the authentication TOKEN passed in the HTTP header and a user_id and the date range values passed in the URL:

```
-H "x-access-token: ${TOKEN}"
?user_id=00077&start_date=2019-11-03&end_date=2019-11-03
```

Response JSON example in case of test success:

```
{
  "ok": 1,
  "res": {"Items": [
    {
      "user_id": "00077",
      "date": "2019-11-03",
      "init": 1572740580000,
      "end": 1572771060000,
      "activity": "sleep",
      "anomaly": "no_anomaly"
    },
    {
      "user_id": "00077",
      "date": "2019-11-03",
      "init": 1572771060000,
      "end": 1572777600000,
      "activity": "kitchen",
      "anomaly": "no_anomaly"
    }
  ]}
}
```

3.2.6 ASAPA Authentication and Authorisation (HMu)

The ASAPA component is developed to offer a Single-Sign-On (SSO) solution for the SHAPES eco-system. It consists of a database and a cloud-native application that offers a REST API interface for the authentication of SHAPES users. It provides users with a PASETO (Platform-Agnostic Security Tokens) token that is globally accepted in the SHAPES eco-system for authentication. The PASETO token expires 10 minutes after issue. The token can be refreshed within the time-frame before expiration.

3.2.6.1 Overview of deployment options

Docker images are available for the deployment of the ASAPA component at a containerised environment (e.g. the pilot's premises). The ASAPA component has been deployed and tested locally, on a Kubernetes cluster through a Continuous Integration/Continuous Deployment (CI/CD) pipeline. The ASAPA API can be accessed through the following URL:

<https://kubernetes.pasiphae.eu/shapes/asapa>

The swagger definition of the API, for security reasons, is not publicly available. Nevertheless, it is provided, upon request, to the relevant partners. Access on the ASAPA RESTful API can be achieved by providing an API-key in **every** request, by adding the "X-Shapes-Key" header, and the API-key value, and the authentication token in the request's headers (see example below).

```
{
  "X-Shapes-Key": "key",
  "X-Pasiphae-Auth": "token"
}
```

3.2.6.2 Interfaces offered

Table 49 – ASAPA's interfaces.

| I/F | Feature | Endpoint/Queue | Description | Produce r/ Resource | Consume r/ Caller |
|------|----------------------|--------------------------------------------------------------|-----------------------------------------------|---------------------------|---------------------------------|
| REST | POST – register user | /auth/register | It registers a new user to SHAPES | ASAPA | Any other SHAPES core component |
| REST | POST – Login user | /auth/login | Logins an existing user | ASAPA | Any other SHAPES core component |
| REST | GET | /realms/<realm_id>/organizations/<org_id>/users/<user_email> | Get data of a user of a specific organization | ASAPA | SymbloTe |

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

| I/F | Feature | Endpoint/Queue | Description | Produce r/ Resourc e | Consume r/ Caller |
|------|---------|------------------|------------------------------------|-------------------------------|---------------------------------|
| REST | GET | /users/<user_id> | Get data of a user with his ID | ASAPA | Any other SHAPES core component |
| REST | GET | /users/me | Get data of current logged in user | ASAPA | Any other SHAPES core component |

3.2.6.3 Test cases and Validation

The health status of ASAPA can be checked at the following endpoint: <https://kubernetes.pasiphae.eu/shapes/asapa/health> with a GET request. The above endpoint can be used to test the full functionality of the ASAPA component.

Table 50 - **Interface asapaAPI-1.** This interface is used for checking the health of ASAPA.

| I/F | Test case | Method | Call | Result |
|------------|--------------|--------|----------------------|-----------|
| asapaAPI-1 | asapaAPI-1.A | POST | /shapes/asapa/health | Pass/Fail |

asapaAPI-1. A. This test case confirms the correct functionality of ASAPA

Response is given in JSON:

```
{
  'The ASAPA service is up and running!'
}
```

The basic use case scenario is the registration of a SHAPES user.

Table 51 - **Interface asapaAPI-2.** This interface is used to register a user to ASAPA.

| I/F | Test case | Method | Call | Result |
|------------|--------------|--------|-----------------------------------------------------------------------------------------------------------------------------------|-----------|
| asapaAPI-2 | asapaAPI-2.A | POST | https://kubernetes.pasiphae.eu/shapes/asapa/auth/register | Pass/Fail |

asapaAPI-2.A. This test case checks if a user is registered to ASAPA successfully. Parameters required:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

```
{  
  "email": "string",  
  "password": "string"  
}
```

Additional data can be included in the request, in the same format as shown below. (e.g. `"Last_name": "smith"`).

```
{  
  "email": "example@gmail.com",  
  "password": "example"  
}
```

Response is given in JSON:

```
{  
  "code": "string",  
  "count": "string",  
  "error": "string",  
  "items": list ["string"],  
  "message": "string",  
  "more": "string",  
  "page": "string",  
  "page_size": "string",  
  "total": "string"  
}
```

For example:

```
{  
  "code": 200,  
  "count": 1,  
  "error": "",  
  "items": [  
    {  
      "_id": {  
        "$oid": "objectID"  
      },  
      "active": true,  
      "created_at": {  
        "$date": 1631546317266  
      },  
      "email": "user_email",  
      "roles": []  
    }  
  ],  
  "message": "REGISTER_USER_SUCCESS",  
}
```



```
"more": "",  
"page": 1,  
"page_size": 20,  
"total": 0  
}
```

3.2.7 Front-end App (EDGE)

The SHAPES Front-end App was developed to provide a unique access point to the SHAPES Digital Solutions running on smartphones or tablets (devices), easing authentication, access and navigation even to users with low technological skills that could exhibit difficulty interacting with novel technologies.

Overall, the SHAPES Front-end App aims:

- To represent the main entry point to the SHAPES Digital Solutions running on smartphones or tablets.
- To provide a mechanism for the single authentication of the user in the SHAPES Platform (as opposed to requiring the user to authenticate when accessing each Digital Solution).
- To deliver a single point of access for the various SHAPES Digital Solutions relevant for specific pilots and deployments. Where possible, a simplified access (single-click) to specific Digital Solutions features is envisaged, to minimise the users' challenges navigating the Digital Solutions.

The SHAPES Front-end App is further described in [1].

3.2.7.1 Overview of deployment options

The SHAPES Front-end App was developed for Android devices, and thus is distributed in the form of an Android Package (APK).

The source-code is open and available for SHAPES partners at the SHAPES GitHub, specifically: <https://github.com/SHAPES-H2020/Front-end-App>.

At the moment, the App has been tested with Android Studio 2020.3.1 for Windows 64.bit. All necessary environment and configuration files are setup once the project is opened in Android Studio.

Currently, the "Front-end App" project in Android Studio is setup to generate the following Apps:

- Front-end App for Pilot 3 led by MOIC
- Front-end App for Pilot 3 led CH
- Front-end App for demonstrations. This App is used in the test cases.
- eCare demonstration App. This App is used in the test cases.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

It is required that the environment where the App is running is able to access the SHAPES ASAPA system for authentication purposes. The ASAPA URL can be set in the “settings.xml” file as follows:

```
<string name="API_BASE_URL" translatable="false">
    https://kubernetes.pasiphae.eu/shapes/asapa/</string>
<string name="TEST_API_BASE_URL" translatable="false">
    https://kubernetes.pasiphae.eu/shapes/asapa/</string>
```

The recommended ways to run and test the SHAPES Front-end App are the following:

- Install to a USB connected mobile phone in developer mode running Android OS above version;
- Install to an android emulator;
- The Android OS version must be above 6.0, but version 10 or 11 is recommended.

In this document, the steps to test the App using an android emulator are described.

The installation of an Android emulator can be done directly via Android Studio using the “AVD Manager”.

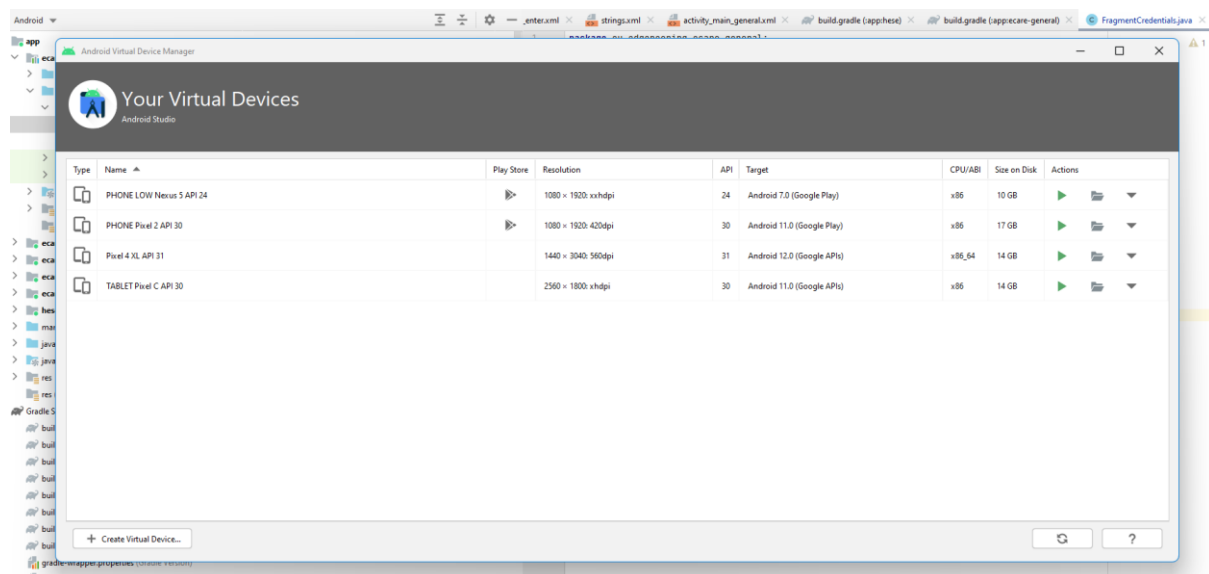


Figure 1- Android Studio AVD Manager

In order to efficiently run the required environment (Android Studio and emulator) a computer with a modern processor and more than 8GB RAM is recommended. Network connection to the ASAPA is also required.

3.2.7.2 Interfaces offered

The following interfaces, described in [1], are tested in this document:

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

- I/F #1 (**Interface between the SHAPES Front-end App and the ASAPA**) that is used to authenticate the user in the SHAPES Platform and SHAPES Digital Solutions. The interface uses ASAPA REST API.
- I/F #3 (**Interface between the SHAPES Front-end App and the SHAPES Digital Solutions**) that is used by the SHAPES Front-end App to launch a SHAPES Digital Solution, as selected by the user.

I/F #2 (Interface between the SHAPES Front-end App and the SHAPES Biometrics Authentication) cannot be tested yet since it depends on biometric authentication services running on the SHAPES Gateway that are not yet available.

3.2.7.3 *Test cases and Validation*

Integration testing is not applicable to the Front-end App. Only functional testing can be applied. Validation tests that ensure the functionality of the app are described below.

In order to run the test cases, a set of initial conditions is required:

1. Android Studio is running and the “Front-end App” project is open.
2. The Android emulator is running.
3. Android Studio is connected to the Android emulator.

Case-App-01: This test case verifies if the App is successfully installed and started in the mobile phone.

Specific Requirements:

- The App has not yet installed

Steps:

1. Select “run app.pt003-moic-demo”

Result

- The build process starts and concludes successfully
- The App is installed in the emulator and is started in the login screen.



Figure 2 - SHAPES Front-end App Login Screen

Case-App-02: This test case verifies if the App successfully authenticates in ASAPA. This case tests I/F #1.

Specific Requirements:

- The App is open in the login screen

Steps:

1. Enter valid login credentials (username and password)
2. Press “Login”

Result

- The App provides a welcome message (i.e., welcome screen)
- The buttons “Enter” and “Logout” appears

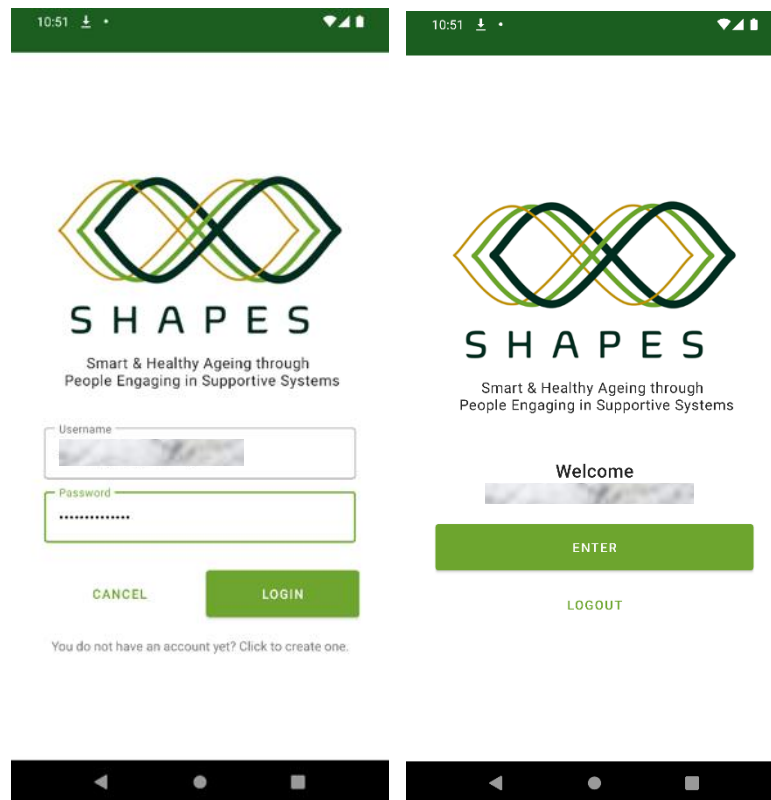


Figure 3 - Successful login in the SHAPES Front-end App

Case-App-02.1: This test case verifies that wrong credentials will not allow entering the App and will generate an error. This case tests I/F #1 and is a special case of Case-App-02.

Specific Requirements:

- The App is open in the login screen

Steps:

1. Enter invalid login credentials (wrong username and/or password)
2. Press “Login”

Result

- The App shows an error message.
- The button “Login” remains active.

Case-App-03: This test case verifies the App navigation functions: enter main screen and go back to login screen

Specific Requirements:

- The App has a valid user and is open in the welcome screen

Steps:

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

1. Click on “Enter”

Result

- The App opens the main screen

Steps-2:

2. Click on the back button (bottom part of the screen)

Result-2:

- The App returns to the welcome screen



Figure 4 - SHAPES Front-end App Main Screen

Case-App-04: This test case verifies the Front-end App navigation functions in opening the eCare App (demo version). It is noted that this case demonstrates the generic mechanism implemented in the Front-end App to open another SHAPES App. This case tests I/F #3.

Specific Requirements:

- The eCare demo App must be installed (select “run ecaredemo”)
- The Front-end App is open in the main screen.

Steps:

1. Click on “Weight”

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

Result

- The eCare demo App opens indicating the username and the destination screen (FRAGMENT_MEASURE_WEIGHT).

Steps:

2. Click on back to return to the Front-end App
3. Click on “Heart Rate”

Result

- The eCare demo App opens indicating the username and the destination screen (FRAGMENT_MEASURE_HEART_RATE).

Case-App-05: This test case verifies the Front-end App in logging out the user.

Specific Requirements:

- The Front-end App is open in the welcome screen.
- The App has a logged user.

Steps:

1. Click on “Logout”

Result

- The user is logged out. The Front-end App shows the “Login” button. The user cannot login unless it authenticates again in ASAPA (see Case-App-02)

3.2.8 Marketplace (HMU)

The SHAPES marketplace allows SHAPES users and third parties to purchase and promote SHAPES services/solutions and products. It consists of three main parts: the frontend, the backend and the marketplace database. The frontend enables SHAPES users to create, list, or purchase SHAPES products, solutions, and services. It completes the business logic for the whole project, through web-based user interface. The marketplace backend utilizes the ASAPA API to authenticate SHAPES users and allow them to use the marketplace. The marketplace contains information about products/services, orders, users, files (e.g., product images/videos, user avatars, etc.), and documentation.

3.2.8.1 Overview of deployment options

The marketplace component is built following the cloud-native applications standards and can be deployed as a cloud-service on any UNIX / Linux / Windows Server with Docker or in any Kubernetes Cluster.

The minimum required hardware capabilities are: 4 CPUs, 4GB of RAM and minimum 50GB of storage space. These values are subject to change, depending on access traffic, load, and storage requirements.

Software requirements are: An instance of Docker service in any operating system or a Kubernetes Cluster. The Marketplace component is deployable with a “Docker-compose” file or a “YAML” file respectively. The Docker images can be built with a “Dockerfile” included on the source-code.

3.2.8.2 Interfaces Offered

The marketplace component does not provide any interfaces for integration with other SHAPES core components.

3.2.8.3 Test cases and Validation

For testing, monitoring, and validating the SHAPES marketplace, we built one endpoint for the marketplace backend that illustrates the health of the backend component and tests the full functionality of the marketplace.

Table 52 - The endpoint of backend health.

| Method | Headers | Endpoint | Body (JSON) | Query Params |
|--------|---------|----------|-------------|--------------|
| GET | --- | /health | --- | --- |

The expected successful response is:

```
{
  "code": "200",
  "message": "Health endpoint of marketplace back-end: SUCCESSFUL",
  "data": { "items": [] },
  "date": "2021-09-10T07:22:07.408Z"
}
```

and the expected failed response is:

```
{
  "code": "500",
  "message": "Health endpoint of marketplace back-end: FAILED",
}
```

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.


```

"data": { "items": [] },
"date": "2021-09-10T07:22:07.408Z"
}

```

Functional testing can be additionally applied to the SHAPES Marketplace component to validate its proper functioning.

For the testing, users can login to marketplace with SHAPES ASAPA credentials. The Login Page is initially shown when accessing the Marketplace URL: https://<deployment_url>/shapes/marketplace-frontend/login.

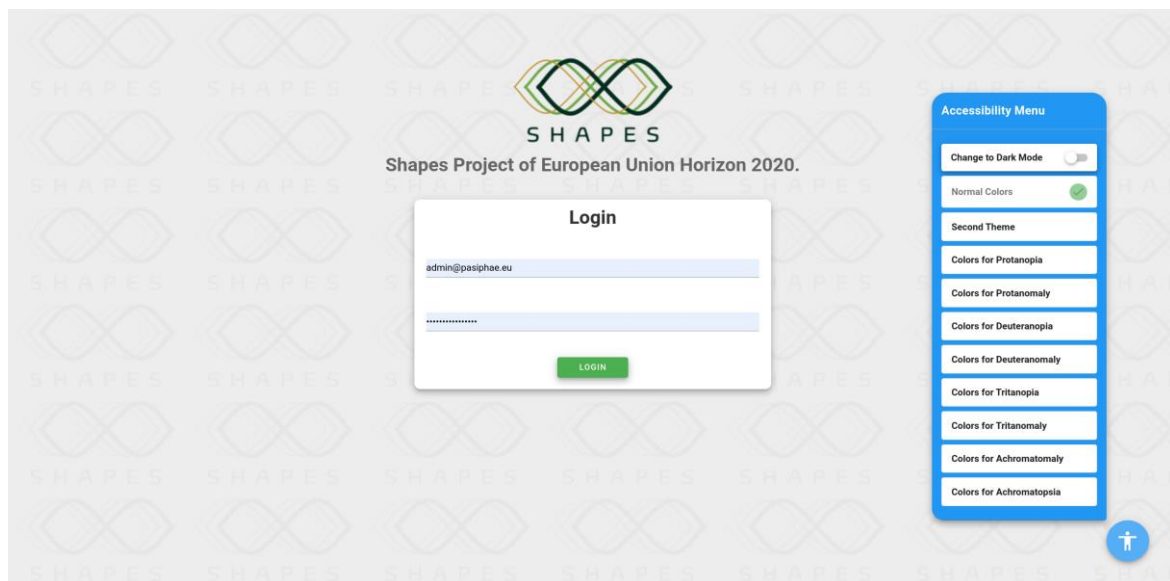


Figure 5 – SHAPES Marketplace Login page.

On the bottom right, there is an accessibility button that opens the accessibility menu for people with special needs. This menu can be accessed at any time from all sub-pages of the Marketplace.

After successful users' login to the Marketplace, a preview of popular and latest products is shown to the Home Page. Users can navigate to other pages from the side-bar on the left.

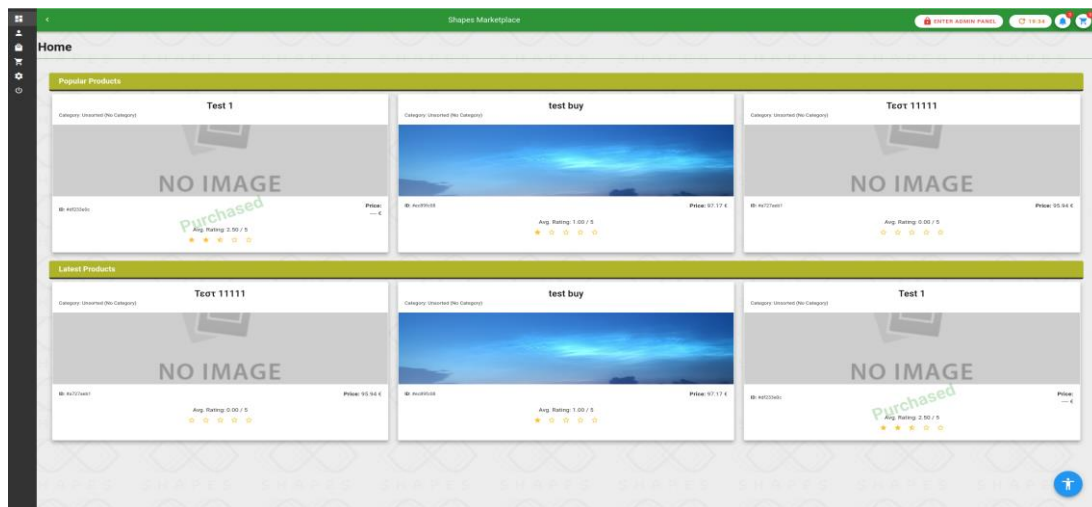


Figure 6 – Marketplace's homepage after successful login.

Navigating to Products page, users can see a list of products that already exist in marketplace.

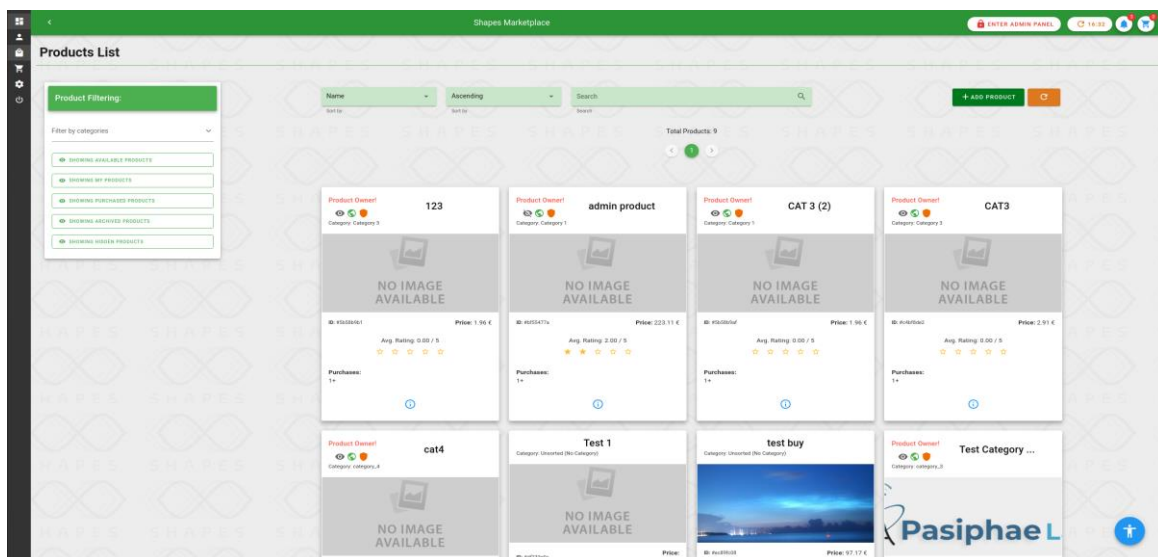
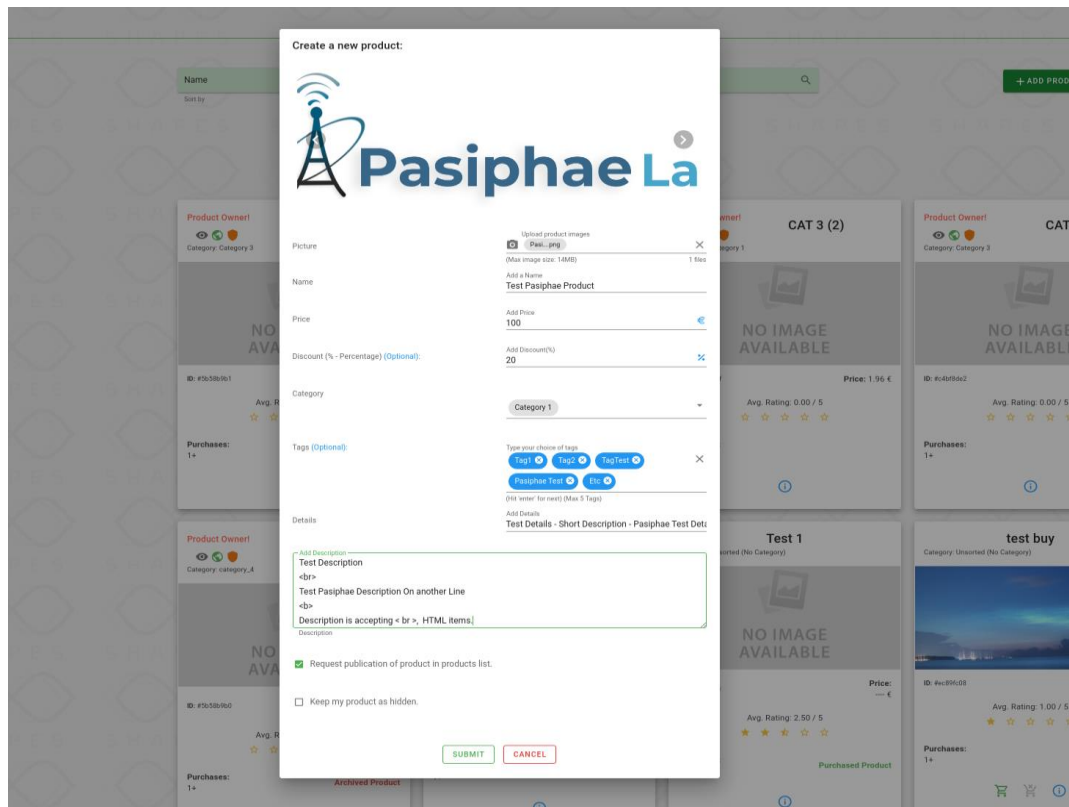


Figure 7 – Marketplace Product list

Additionally, they can add their own product for sale by clicking the top-right green button "Add product". This will open a form for adding details and images for the product that users want to sell.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.



Create a new product:

Pasiphae La

Picture: (Max image size: 14MB) 1 file

Name:

Price: €

Discount (% - Percentage) (Optional): %

Category:

Tags (Optional): (No more for now) (Max 5 tags)

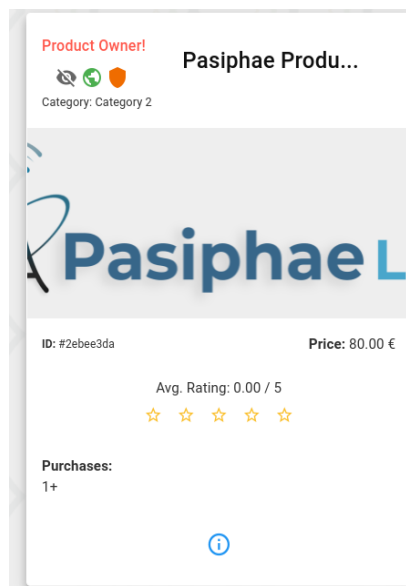
Details:

☒ Request publication of product in products list.

☐ Keep my product as hidden.

Figure 8 – Marketplace: create new product.

After adding a product for sale, it will stay in **pending validation** state until an Administrator approves it. The added product is shown in product list.



Product Owner! **Pasiphae Produ...**

Category: Category 2

Pasiphae L

ID: #2ebee3da Price: 80.00 €

Avg. Rating: 0.00 / 5

Purchases: 1+

Figure 9 – Marketplace Product List.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

The top-left corner is visible to the product owners (creators) only. This showcases the product state, for example orange shield is pending administrator validation. The explanation for each state is visible by hovering the symbol in question.

By pressing the information button (i) on the bottom of a product card. The user can navigate to product information page and if the user is an owner, he/she will be able to edit the product as shown below.

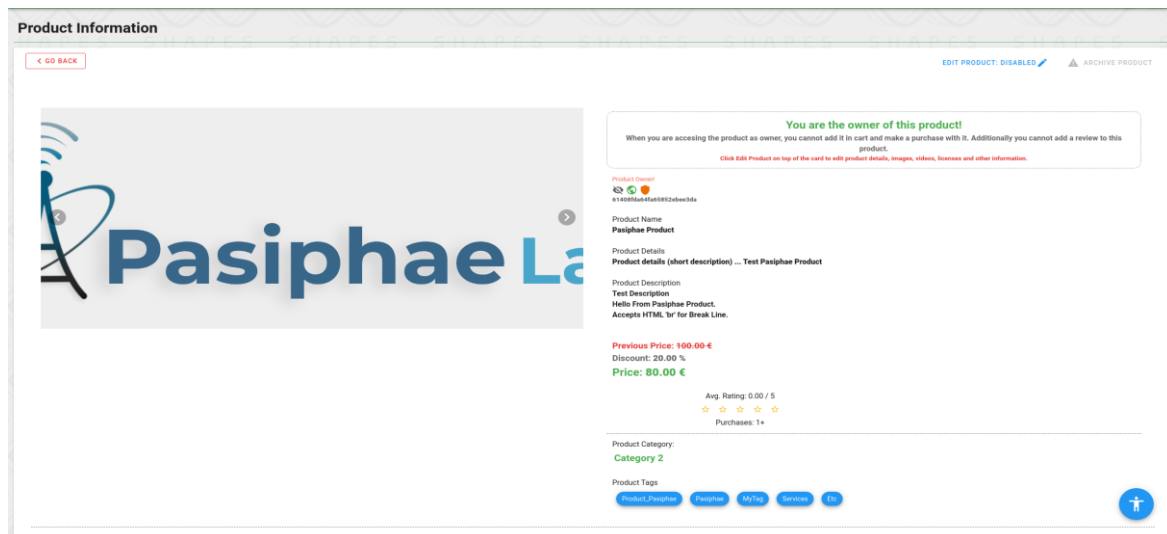


Figure 10 – Marketplace's product information page.

Scrolling down the product information page, users can see product videos (if any), product documentation (this subcomponent is under construction), product licenses (only in case users have purchased the product or users are owners of the product listed – this subcomponent is under construction) and finally in this page users can see the global product's reviews. Additionally, users can add a Review for a product (only in case product is purchased) (this subcomponent is still under construction).

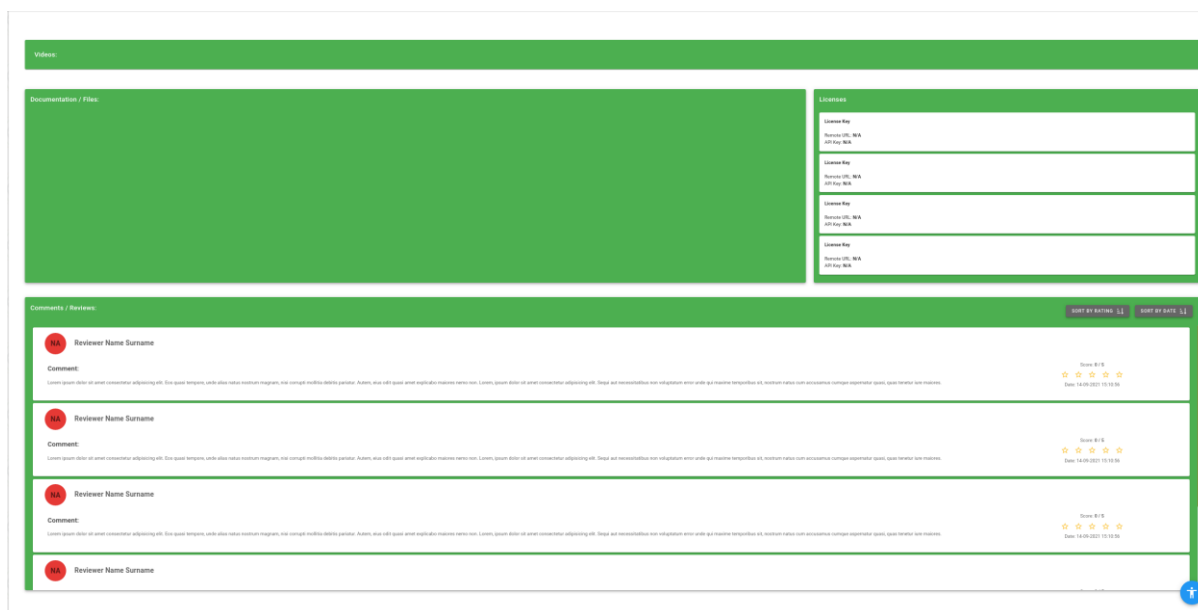


Figure 11 – Marketplace: additional production information details.

For purchasing a product, the scenario is as follows:

1. Users select the products' list or product information page and press “Add In Cart” button as shown in figures bellow.

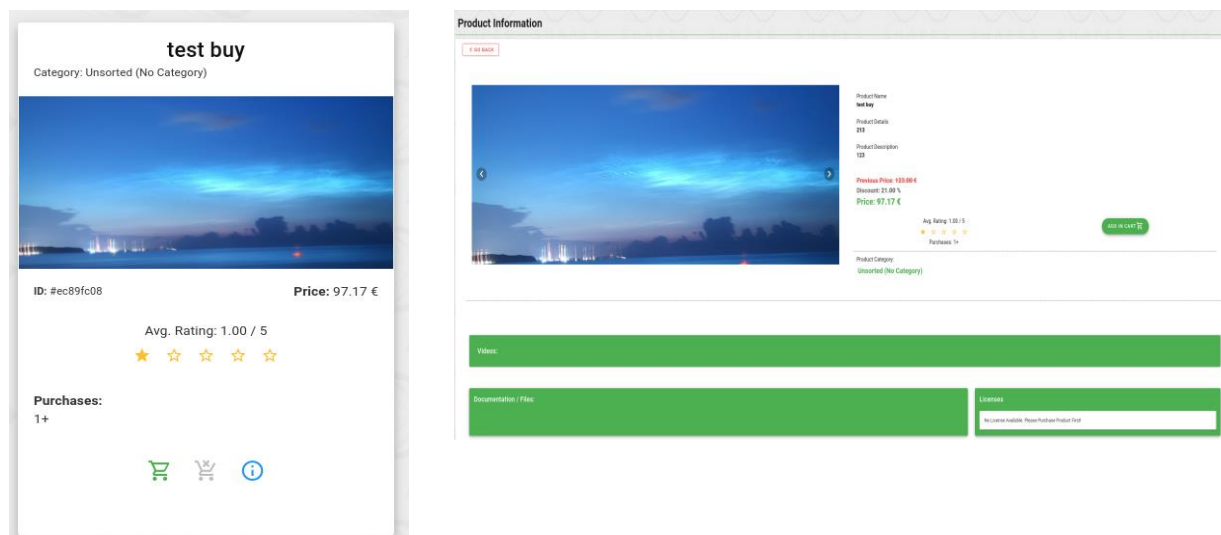


Figure 12 - Marketplace: product purchase scenario (step 1).

These buttons are only visible to products that user is not a “creator / owner”. After adding a product to cart, the user can see the product in the cart (top-right corner) or navigate to cart from sidebar on the left.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

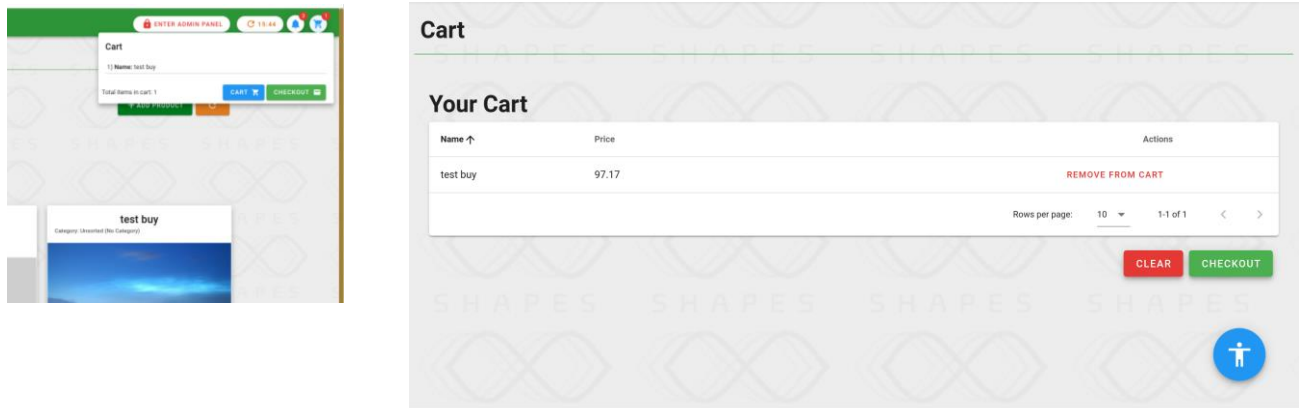


Figure 13 - Marketplace: product purchase scenario (step 2).

2. After navigating to the cart page, users can change/modify products in cart list or clear cart and start over. Moreover, users can navigate to checkout page.
3. Navigating to checkout page will result in execution of a series of eligibility checks for users, before granting eligibility to purchase products. Such eligibility checks are “Complete profile”, “User Verified by Admin”, etc. Users are always informed for the process of eligibility with messages on the checkout page.

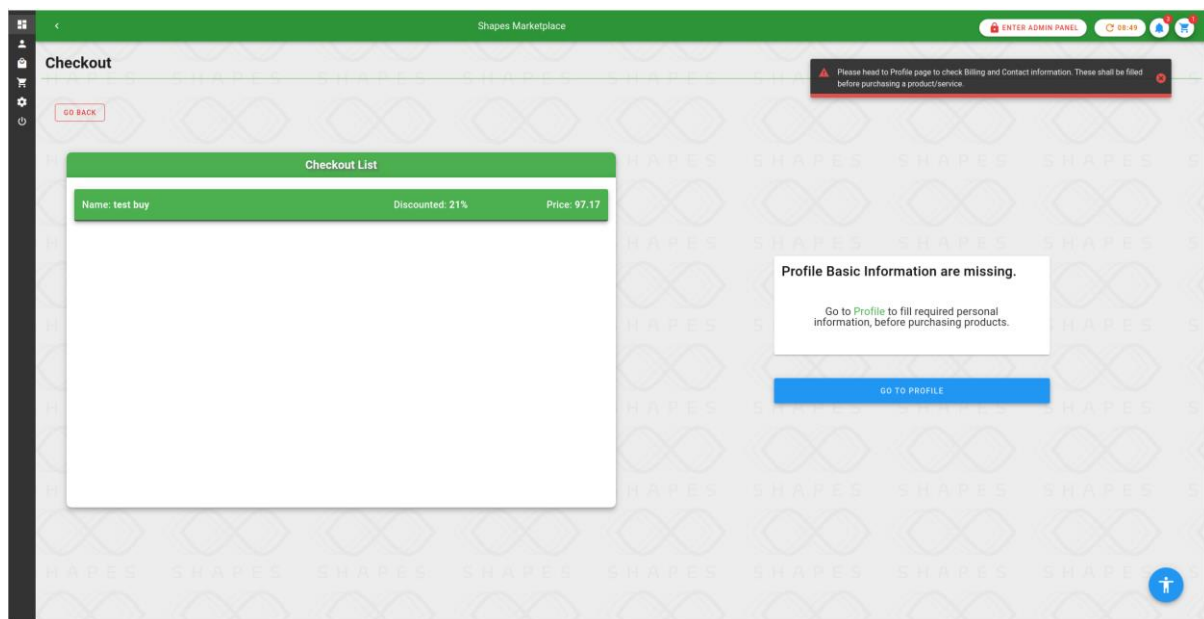


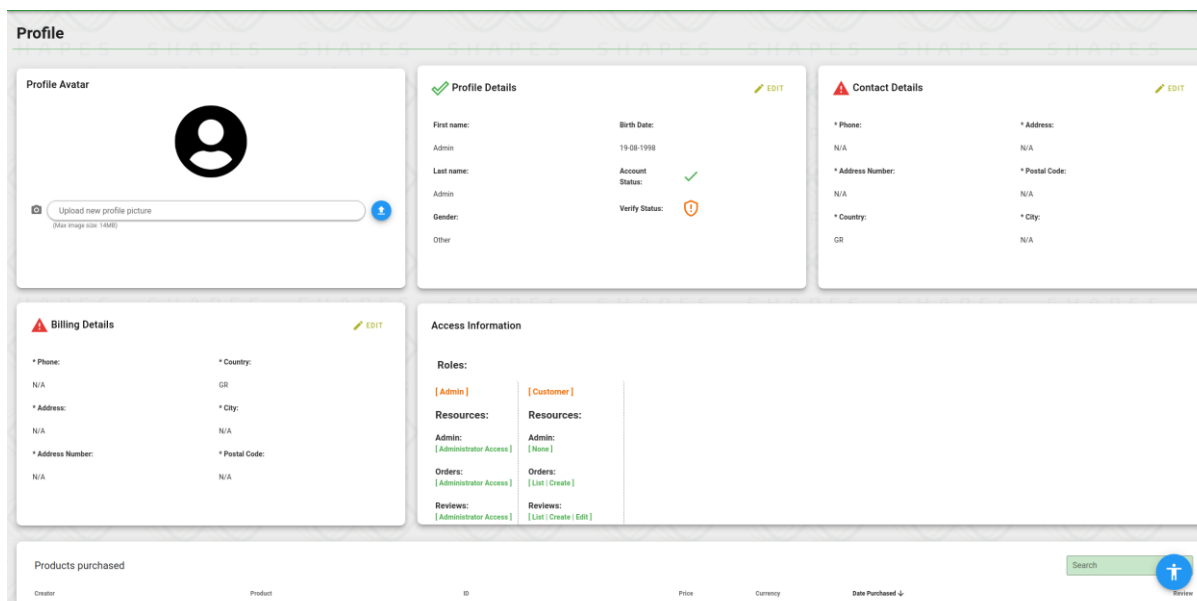
Figure 14 - Marketplace: product purchase scenario (step 3).

In this example users are prompted to complete the basic information on their profile. Users can navigate to profile, by using the blue button below the message, either by using the sidebar on the left.

4. By navigating to the Profile page, users can see various details such as contact details, profile details, billing details, products purchased, profile picture and access

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

roles / resources. By default, every user of ASAPA is granted the CUSTOMER role and only one ADMIN User exists after the initialization of the marketplace component. In the figure below, users can see all information mentioned above and status concerning any information. In this figure we can see “warning” triangles on the left of incomplete profile details. Adding the missing details will result in a pending “Verify Status” state, as shown in middle of Profile Details with a shield icon and check marks on top of every completed detail card. The tooltip of each icon and symbol explanation is triggered on hover of the icon/symbol.



Profile

Profile Avatar

Upload new profile picture (Max image size: 1MB)

Profile Details

First name: Admin Birth Date: 19-08-1998

Last name: Admin Account Status: ✓

Gender: Other Verify Status: ⚠

Contact Details

* Phone: N/A * Address: N/A

* Address Number: N/A * Postal Code: N/A

* Country: GR * City: N/A

Billing Details

* Phone: N/A * Country: GR

* Address: N/A * City: N/A

* Address Number: N/A * Postal Code: N/A

Access Information

Roles:

[Admin] [Customer]

Resources:

Admin: [Administrator Access] Admin: [None]

Orders: [Administrator Access] Orders: [List | Create]

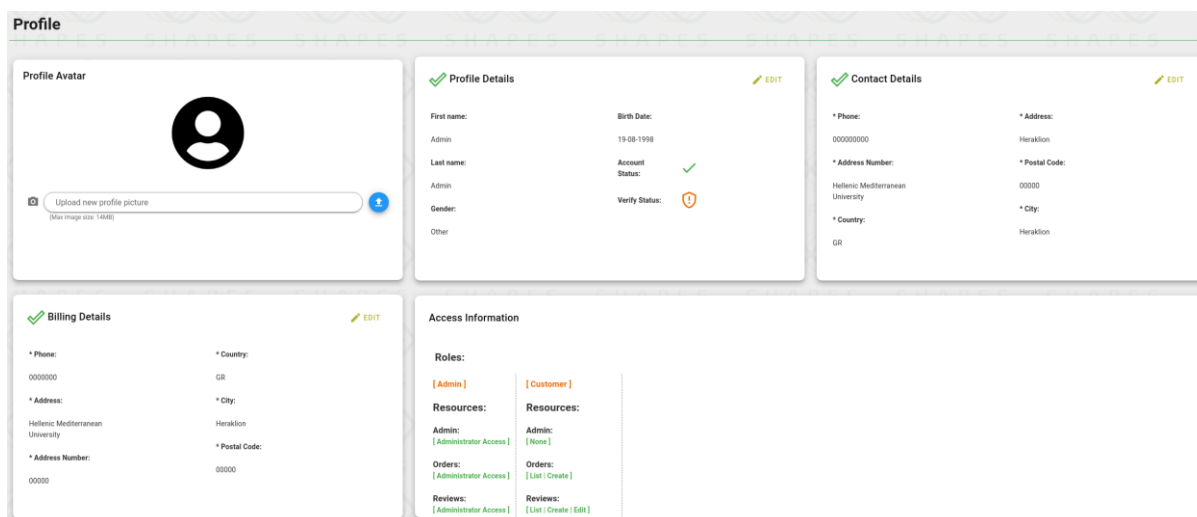
Reviews: [Administrator Access] Reviews: [List | Create | Edit]

Products purchased

Creator Product ID Price Currency Date Purchased

Search [User Icon]

Figure 15 – Before filling profile details.



Profile

Profile Avatar

Upload new profile picture (Max image size: 1MB)

Profile Details

First name: Admin Birth Date: 19-08-1998

Last name: Admin Account Status: ✓

Gender: Other Verify Status: ⚡

Contact Details

* Phone: 000000000 * Address: Heraklion

* Address Number: 00000 * Postal Code: 00000

* Country: GR * City: Heraklion

Billing Details

* Phone: 00000000 * Country: GR

* Address: Hellenic Mediterranean University * City: Heraklion

* Address Number: 00000 * Postal Code: 00000

Access Information

Roles:

[Admin] [Customer]

Resources:

Admin: [Administrator Access] Admin: [None]

Orders: [Administrator Access] Orders: [List | Create]

Reviews: [Administrator Access] Reviews: [List | Create | Edit]

Figure 16 - After filling profile details.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

5. After an administrator approves profile details, users will be able to purchase products in the marketplace. The “Verified” Status is indicated in user’s profile as:

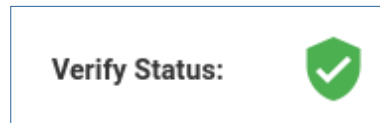


Figure 17 – Verify Status

6. Users can then navigate to checkout page, one more time and after verification, they will be able to see the credit card form as shown in the figure below. The users will be able to add their credit card data which will be automatically validated and securely sent to “Stripe” (external payment service) backend.

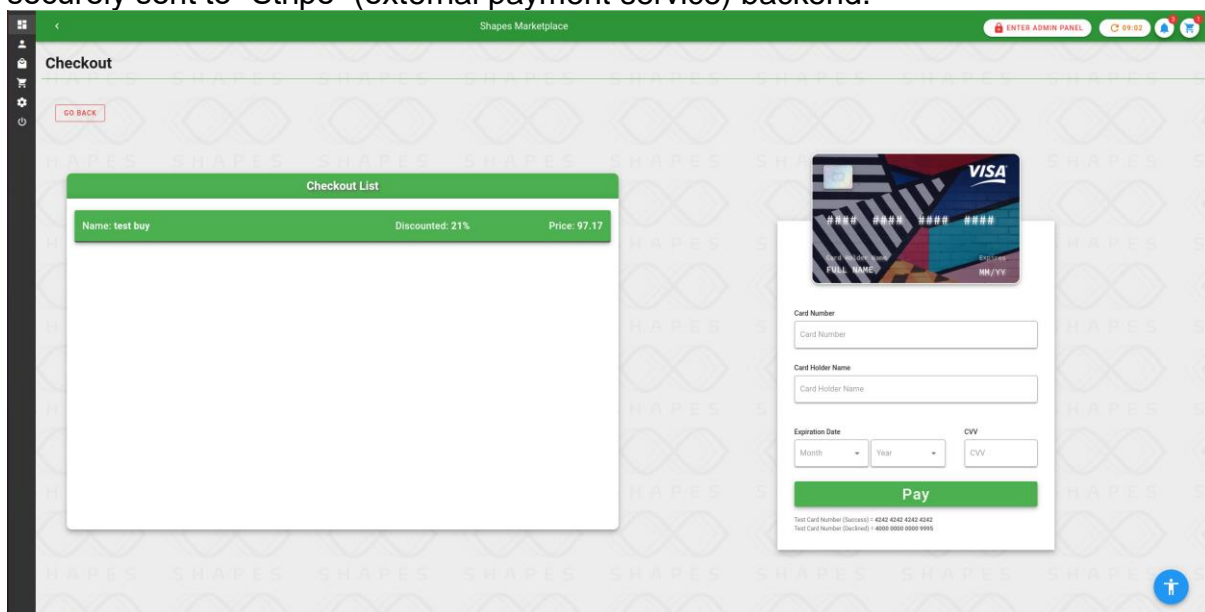


Figure 18 – Marketplace checkout page.

7. Finally, after the successful payment, users are redirected to the home page. Additionally, users can see the purchased products in the product list, the profile page and the home page (if product is popular or latest) as figures bellow:

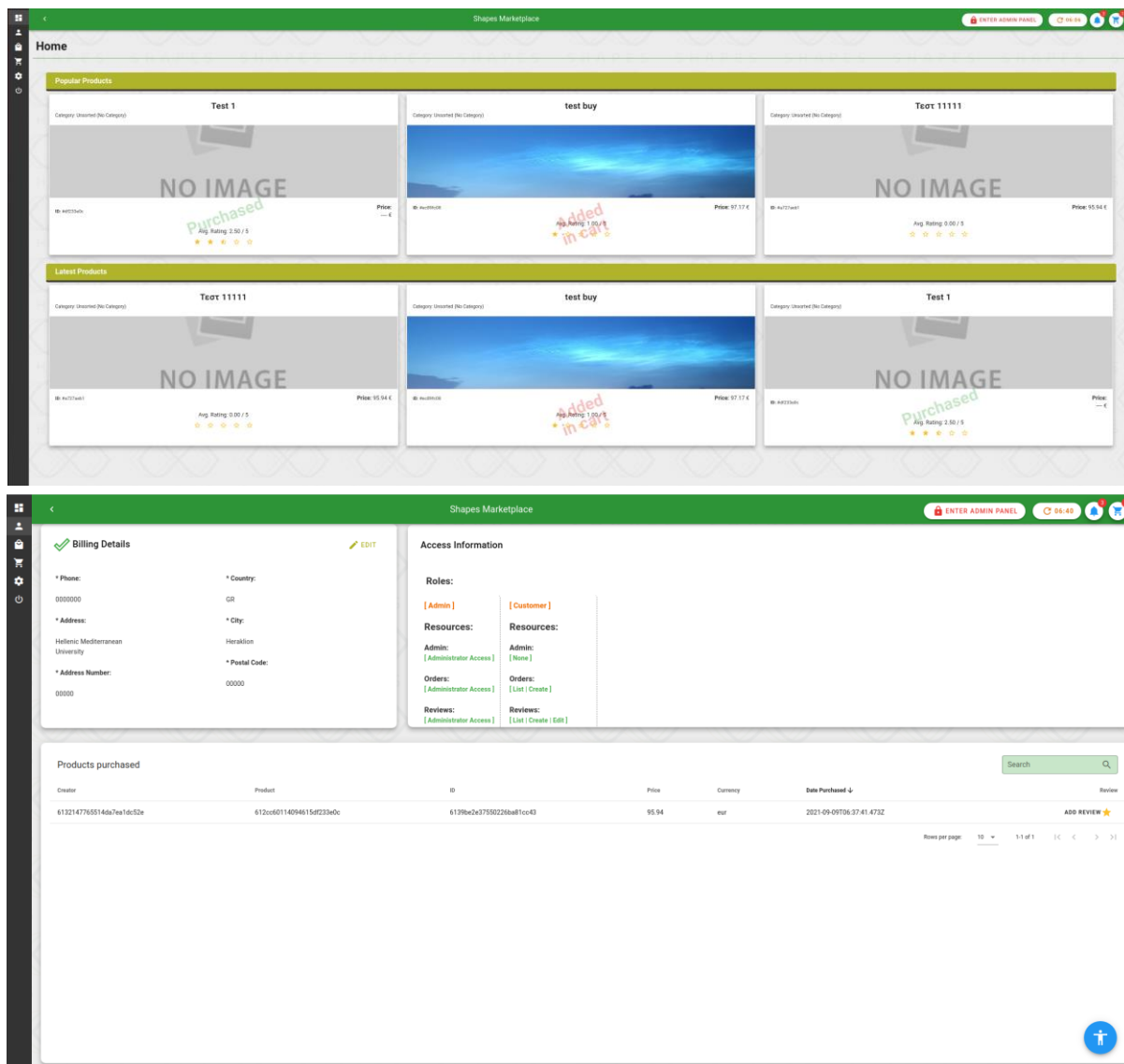


Figure 19 – SHAPES Marketplace homepage after successful registration of a new product.

4 Results and Conclusions

This deliverable summarises and details the work performed in WP4 until M24. Although originally spinning of our Task 4.8, it has been extended with contributions also from other tasks in WP4 thus to provide a comprehensive view of the current state of the implementation and integration of the SHAPES Technological Platform, its testing and deployment, demonstrating its readiness for integration with Digital Solutions in cooperation with WP5 to get the whole SHAPES framework ready for deployment and application in field trials under WP6.

The complete list of contribution WP4 tasks includes:

- **Task 4.3** “Implementation of Mediation Framework & Interoperability Services”
- **Task 4.4** “Implementation & Deployment of Secure Cloud & Big Data Platform”
- **Task 4.5** “Human Interaction & Visual Mapping”
- **Task 4.6** “SHAPES Authentication, Security & Privacy Assurance”
- **Task 4.7** “SHAPES Gateway Reference Implementation”
- **Task 4.8** “Integration and Testing of SHAPES TP”

This report provides a detailed description of the integration and testing approach used to integrate the SHAPES core components into a working SHAPES TP prototype. It describes the integration framework used, provides information regarding the deployment requirements and options for each of the SHAPES core components and finally describes the test cases for the integration or functional testing to validate their proper functioning. Additionally, digital solutions such as from VICOM, UCLM, SciFy, Omnitor and MedSyn have provided a description pertaining to the deployment of their solutions and the test cases they have provided, to test their interfaces where applicable.

This deliverable will be further updated in month 30 or in May 2022 with the new version (*D4.4 Integration Testing Results, preliminary version*), focusing on the integration outcomes of WP4 in the upcoming six months, as the various modules of the SHAPES TP (both core components and digital solutions) are deployed and tested in different pilot sites across Europe, according to the demo scenarios of each one of them.

5 Ethical Requirements Check

The focus of this compliance check is on the ethical requirements defined in D8.4 and having impact on the SHAPES solution (technology and related digital services, user processes and support, governance-, business- and ecosystem models).

Table 53 – Compliance check on ethical requirements.

| Ethical issue (corresponding subsection of D8.4 in brackets) | How it has been taken into account in this deliverable (if relevant) |
|--------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Fundamental Rights (3.1) | N/A (no private identifiable data exchanged or processed by SHAPES-TP) |
| Biomedical Ethics and Ethics of Care (3.2) | N/A (no private identifiable data exchanged or processed by SHAPES-TP) |
| CRPD and supported decision-making (3.3) | N/A (Decision Support System is not provided by SHAPES-TP) |
| Capabilities approach (3.4) | N/A |
| Sustainable Development and CSR (4.1) | N/A to SHAPES-TP (covered by Digital Solutions) |
| Customer logic approach (4.2) | SHAPES-TP follows service approach adopted by Digital Solutions and complies with their CLA strategy |
| Artificial intelligence (4.3) | Analytics Engine is the only one employing AI technologies. It is fully user agnostic, privacy preserving by dealing with non-identifiable data, excluding any personal information. Relevant for the data analytics solutions. Please see section 3.2.5 and Appendix 1. |
| Digital transformation (4.4) | N/A (SHAPES-TP does not provide services, merely facilitates interoperable data exchange among service platforms) |
| Privacy and data protection (5) | N/A (no private identifiable data exchanged or processed by SHAPES-TP, irrespective of this encrypted data exchange with dedicated authentication is applied to core components, while authentication of access to user data is governed by Digital Solutions) |

| | |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cyber security and resilience (6) | SHAPES-TP employs a number of cyber security protections, from secure access from external Digital Solutions using IP/MAC access control, using encrypted communication interfaces and most importantly making sure that NO private identifiable user data is exchanged or processed with or by core components. |
| Digital inclusion (7.1) | N/A |
| The moral division of labour (7.2) | N/A |
| Care givers and welfare technology (7.3) | N/A |
| Movement of caregivers across Europe (7.4) | N/A |

6 References

- [1] SHAPES Consortium (2020). D4.1 – SHAPES Technological Platform (TP) Requirements and Architecture, May 2021.
- [2] SHAPES Consortium (2020). D5.3 – SHAPES Digital Solutions V2 – October 2021.

Annex 1 Digital Solutions – Integration Efforts

1.1 NOT!FY Digital Solution (OMN)

Omnitor **NOTiFY** is a cloud-based platform currently used to make citizens aware of incoming Total Conversation calls. Omnitor has developed two different versions of NOTiFY, the NOTiFY Smart plug and NOTiFY v1. NOTiFY smart plug consist of a built-in relay that can remotely be switched on/off and offers energy monitoring capability. NOTiFY V1 consists of four exposed relays that external alerting devices can be connected to.

1.1.1 Overview of deployment options

The server is temporarily deployed at AWS (Amazon Web Service). This has been developed to offer a set of Rest API services to cover the needs for Big Data Companies to create an algorithm and analyse data.

The inbound REST API for the server has been implemented using HTTP POST and getting the values from the NOTiFY Smart plug while the requesting data from the server could either be HTTP POST or GET, however, POST is recommended. The temporary server offers a set of Rest API.

This is a temporary solution and thus no documentation has been created.

1.1.2 Interfaces offered

There are two interfaces offered, the first one is the “Inbound Data from Smart plug” which is only used for the Smart plugs to upload data.

The second one is “request data from the server”, this is used mainly but not exclusively by the Big Data Companies to gather the needed data, any partner that needs to access the data could use the same Rest API.

Rest API specifications are presented below:

The authentication is not presented here and could be requested from Omnitor.

Inbound Data from Smart Plug

The inbound data is the data that the smart plug is POSTING to the server every 30 – 31 seconds.

Table 54 – Notify interfaces.

| I/F | Feature | Endpoint/Queue | Description | Producer/Resource | Consumer/Caller |
|------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------|
| REST | POST | POST /default/uploadNotifyData HTTP/1.1 Host: 9d7upn31kd.execute-api.eu-north-1.amazonaws.com Authorization: \$PASSWORD Content-Type: application/json Content-length: varies, depending on data HTTP data: { "boxid": "unique to the device", "current": AB.XYZ, "voltage": AB.XYZ, "power": AB.XYZ, "temp": XY.Z } | Register the data from the smart plug. The registered data are: Boxid, current, voltage, power and device temperature | Omnitor API | NOTIFY Smart Plug |

Table 55 – Notify: Request data from the server.

| I/F | Feature | Endpoint/Queue | Description | Producer/Resource | Consumer/Caller |
|------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------|-------------------|------------------------------------------------|
| REST | POST | POST /default/uploadNotifyData HTTP/1.1 Authorization: \$PASSWORD Host: 9d7upn31kd.execute-api.eu-north-1.amazonaws.com Content-Type: application/json Content-length: varies, depending on data HTTP data: { "startTime": "YYYY-MM-DD hh:mm:ss", "endTime": "YYYY-MM-DD hh:mm:ss" } | Accessing the uploaded data between the time intervals. | Omnitor API | Big Data Platform or any other SHAPES partner. |

1.1.3 Test cases and Validation

Table 56 – OmnitorAPI test cases.

| I/F | Test case | Method | Call | Result |
|------------|-----------|--------|-------------------------------------------------------------------------------|-----------------|
| OmnitorAPI | OUTPUT | POST | https://27osqql772.execute-api.eu-north-1.amazonaws.com/default/getNotifyData | OK/Unauthorized |

Test parameters:

```
{
  "OmnitorToken": "string"
  "startTime": "Date"
  "endTime": "Date"
}
```

Response is given in JSON:

```
{
  "id": "Numbers",
  "boxid": "String",
  "current": "Numbers",
  "voltage": "Numbers",
  "power": "Numbers",
  "temp": "Numbers",
  "time": "Date"
}
```


1.2 FACECOG Tool to Support User Authentication and for People Identification at a Distance (VICOM)

FACECOG (Face Recognition Solution for Heterogeneous IoT Platforms) API version 1.0 is a software module from Vicomtech's Viulib library (<http://www.viulib.org/>), designed to analyse facial identities in the contexts of user verification and person identification in heterogeneous IoT platforms.

It can support the user authentication process based on user/password, to make it more user-friendly, but also for the recognition of potential users at a distance (to ask for attention, etc).

FACECOG has been designed to overcome the problem of effectively deploying face recognition algorithms in the high variety of edge devices and robots that could be part of an IoT platform, like SHAPES. It includes an encryption mechanism to avoid compromising the privacy of users, based on fully homomorphic encryption, which allows maintaining the biometric data always encrypted, even during matching operations.

Further details of FACECOG are included in [2].

1.2.1 Overview of deployment options

- FACECOG is deployed as a server in a local machine with sufficient computing capability for Deep Neural Network (DNN) inference, and not in the cloud. shows the workflow for the deployment of FACECOG. There might be several kinds of devices and robots with whom users might interact. This setup considers an IoT edge gateway with sufficient computing capabilities to run the DNNs of FACECOG to which the devices not suitable for this computation would send recognition requests to process captured images and receive the corresponding responses. FACECOG would be deployed on this gateway as containerized services to handle multiple requests at once, but also on the devices and robots with sufficient computing capabilities, to reduce the latency as much as possible in all cases. To allow users to register in one device and be recognized by another, the biometric data should be shared among the devices where FACECOG is deployed. Even if the network is private, the data would be encrypted, and an administrator would manage the encryption keys to preserve privacy with a secure element.

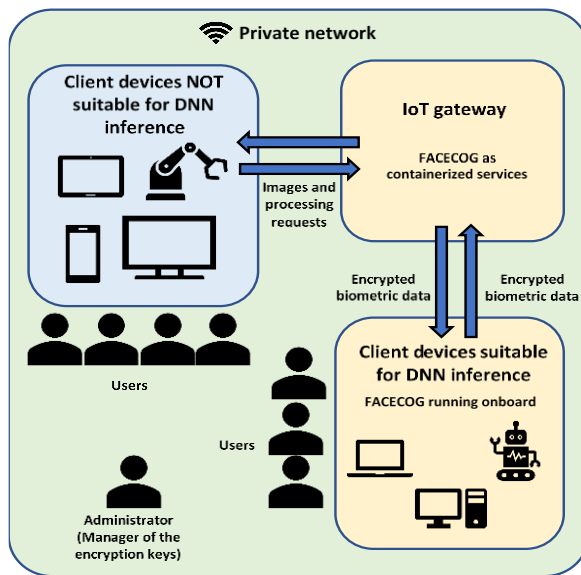


Figure 20 - Deployment diagram of FACECOG.

- The following Docker image needs to be installed (Vicomtech gives the access to this image), like this:

```
$ docker pull quay.io/vicomtech/shapes:facecog_v1
```

- A batch file is also provided to run the FACECOG server in the local machine (robot, gateway, PC), with the following code (Linux version):

```
#!/bin/bash

# docker run --runtime nvidia --rm -p 8000:8000 facecog:latest
# /facecog/run_services.py
argn=$#

echo "$# args"

if [ $argn -le 3 ]; then
echo "Usage: $ $0 <pub_key_storage_path> <pri_key_storage_path>
<galois_key_storage_path> <relin_key_storage_path>"
else
dockername=facecog:latest
pub_key_storage_path=$1
pri_key_storage_path=$2
galois_key_storage_path=$3
relin_key_storage_path=$4

docker run --runtime=nvidia \
--rm \
--net=bridge \
-v $pub_key_storage_path:/key_storage/pub/ \
```

```
-v $pri_key_storage_path:/key_storage/pri/ \
-v $galois_key_storage_path:/key_storage/galois/ \
-v $relin_key_storage_path:/key_storage/relin/ \
-p 8000:8000 \
-e FACECOG_MODEL_PATH=/facecog/lib/viulib/ \
--entrypoint "/usr/bin/python3" \
--name=facecog-services \
$dockername \
run_services.py --reload --port 8000 --
pub_key_storage_path=/key_storage/pub/ --
pri_key_storage_path=/key_storage/pri/ --
galois_key_storage_path=/key_storage/galois/ --
relin_key_storage_path=/key_storage/relin/
fi
```

- As it can be seen in this code, the installer needs to include the paths where the 4 keys used to handle the biometric data with security are stored. The keys are the public key, the private key, the Galois key and the relinearization key.
- FACECOG can be deployed in machines with ARM64 architecture (like Jetson TX2, Jetson Nano, Raspberry Pi 4, etc) and 2GB of RAM memory or more. We recommend including in the machine a secure element, like a Trusted Platform Module (TPM) or a Trusted Execution Environment (TEE) to store the private key. TPMs are normally available in modern computer PC mother boards (since 2016). TEEs are available in Jetson Xavier NX, Jetson AGX Xavier series, and Jetson TX2 series devices.

1.2.2 Interfaces offered

The system services include the needed tools and features to manage user registration, identification, and search scenarios with the HTTP REST protocol.

The images sent to the services must be encoded using base64 encoding and appended to the service arguments as a string or a list of strings depending on the case.

The general flow of the identification and verification use cases is:

For user registration

- 2 Detect faces in the image (*getAlignedFaces*)
- 3 Evaluate if face position is correct (*evaluateFacialPose*)
- 4 If face pose is correct add user to database (*addUserData*)

For verification (once user was registered)

1. Detect faces in the image (*getAlignedFaces*)
2. Evaluate if face position is correct (*evaluateFacialPose*)
3. If face pose is correct check user id and image similarity (*verifyUser*)
4. Check if verification process detects spoofing attempts

For identification (once user was registered)

1. Detect faces in the image (*getAlignedFaces*)
2. Evaluate if face position is correct (*evaluateFacialPose*)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

3. If face pose is correct check user in database (*identifyUser*)

User registration

You can add and remove users from the database. For the user registration, the user image should be properly aligned to reduce the differences in a posterior identification process.

Get aligned faces (*getAlignedFaces*)

The services include a function to detect faces in an image. It returns a list of detected faces, cropped each to their bounding box and aligned to normalize the face pose.

The arguments to call this service:

```
{
  "captured_image_b64": base64 image as string,
}
```

The response message:

```
{
  "face_imgs_base64": list of faces,
  "face_regions": list of rois,
  "landmarks": list of landmarks,
  "poses" : list of head angles
}
```

Where *aligned_faces_b64* is a list of face images encoded in base64, *rois* is a list of image regions of the detected faces in the original image (in format *[x, y, width, height]*), *landmarks* is a list of a set of facial feature landmarks for the detected faces (one list of *N* landmarks for each detected image) and *poses* is a list of pose values (Euler rotation angles) for each of the detected poses.

All the lists in the result message should have the same length, maintaining the same order between lists (i.e., an element with index *i* in any list corresponds to the data for the face with the same index *i* in the *face_imgs_base64* image list).

Add user to database (*addUserData*)

Adds a user to the identification database. If the specified user id is already present in the database, the call updates the user data for that id.

The arguments to call this service:

```
{
  "user_id": user id as string,
  "aligned_face_img_b64": base64 image as string
}
```

Where *user_id* is the id value (as alphanumeric string) of the registered user and *aligned_face_img_b64* is aligned and cropped face image of the registered user.

This call has no response message.

Remove user from database (*removeUserData*)

Removes the user with the specified id from the database.

The arguments to call this service:

```
{  
  "user_id" : user id as string  
}
```

Where *user_id* is the id of the user to be removed.

This call has not any response message.

Check registered data (*getRegistrationInfo*)

This service returns the list of ids included in the database. This service has no input arguments.

The response message:

```
{  
  "users": list of user ids as strings  
}
```

Where *users* is a list of user ids in the database.

Identification (*identifyUser*)

This service estimates the identity of the user using the aligned face image crop parsed as an argument.

The arguments to call this service:

```
{  
  "aligned_face_img_b64": base64 image as string,  
  "ident_tresh": identification threshold  
}
```

Where *aligned_face_img_b64* is the face image (cropped and aligned) and *ident_tresh* is the similarity value threshold to be overtaken to be set as correct. 0.6 is a good reference value for the threshold.

The response message:

```
{  
  "output": list of identification results [id, score]  
}
```

Where output is a list of two values: *id* (with index 0) representing the estimated id of the user, and *score* (with index 1) given the final similarity value.

In the case of no correct identification, the return message defines the is as Unknown and the score the similarity value of the closest face in the database. This is an example of a return message with no identification:

```
{  
  "output": ["Unknown", 0.2]  
}
```

User verification (*verifyUser*)

This service verifies that the parsed user image and the id correspond with the same individual based on the information for that user in the registration database.

The arguments to call this service:

```
{  
  "user_id": user id as string,  
  "aligned_face_img_b64": base64 image as string,  
  «verif_thresh": verification threshold as float,  
  «liveness_thresh": liveness threshold as float,  
  "spoofing_thresh": spoofing threshold as float  
}
```

Where *user_id* is the expected id of the face image to be validated, *aligned_face_img_b64* is the face image (cropped and aligned), *verif_thresh* is the verification threshold, *liveness_thresh* is the liveness threshold and *spoofing_thresh* is the anti-spoofing threshold.

Good reference values for the thresholds in this call could be:

- *verif_thresh* = 0.6,
- *liveness_thresh* = 0.6,
- *spoofing_thresh* = 0.6

The response message:

```
{  
  "output": list of identification results for verification, liveness and  
  spoofing analysis.  
}
```

output parameter includes the information about three different aspects of this analysis (user verification, user liveness and spoofing attempt, in this order). In each case, it includes a Boolean flag and a score value (from 0 to 1).

If the verification flag is True, the parsed user id matches with the user in the image. If the liveness flag is True, the user image corresponds to a real person and not to an image, and if spoofing is True, the system estimates that there is a spoofing attempt.

An example output message could be:

```
{
  "output": [
    [True, 0.8], # User validation success with score 0.8
    [True, 1], # User liveness is positively validated with score 1
    [False, 0.1] # Spoofing attempt is discarded with score 0.
  ]
}
```

Other tools

Server status check (*serverStatus*)

This service can be used to check if FACECOG is correctly running on the server. It has no input arguments and returns a message with a string showing the current status.

The response message:

```
{
  "status": FaceCog status string
}
```

The value of the status message could be "FACECOG initialized: True" or "FACECOG initialized: False" depending on the status.

Evaluate facial pose (*evaluateFacialPose*)

To improve the user verification and identification processes, the user face should be frontally looking at the camera. To help the user to get a correct pose for the identification, the FACECOG services include an evaluation tool that determines the actions to improve the face image capture.

This service uses the information provided by *getAlignedFaces* service and returns a set of feedback strings to help the user to correct the head pose and location.

The arguments to call this service:

```
{
  "face_region ": face roi in the image,
  "landmarks": list of facial landmarks,
  "angles: " head angles,string",
  "ref_region": an image region where the user face must be located
}
```

The response message:

```
{  
  "feedback": Feedback message as a string  
}
```

The values of the feedback message could be:

- "CORRECT"
- "LOOK STRAIGHT"
- "MOVE CLOSER"
- "MOVE BACKWARDS"
- "MOVE TO THE RIGHT"
- "MOVE TO THE LEFT"
- "MOVE DOWNWARDS"
- "MOVE UPWARDS"

1.2.3 Test cases and Validation

Viulib's FACECOG module has been used in several projects where Vicomtech has the role of face recognition technology provider, so the whole SDK is being continuously tested in several environments.

In terms of Android technical implementation, the aforementioned functionality can be achieved by using Android Intents and Content Providers.

1.3 Adilib Chatbot Building Platform (VICOM)

Adilib is a comprehensive software for the development of chatbots. Composed of different technological modules, it allows the understanding of the text expressed by the user, its contextualisation based on interaction and external information, and the communication of responses in natural language, whether being static or dynamic.

Adilib is a general-purpose software, which allow to create and configure bot for different domains by introducing and annotating user utterances and defining interaction rules.

1.3.1 Overview of deployment options

- To install Adilib, it is necessary to have access to the Adilib-installer, which is a docker compose configuration file. Once downloaded, it is recommended to place the folder on a partition where there is enough free space.
- Adilib has been tested can be deployed in in Ubuntu 16, 18 and 20 LTS environments, and it can be deployed in any machine with the specified OS and hardware requirements.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

- For SHAPES, Adilib is being served as a Software as a Service on an OVH cloud, based in France.

Software requirements:

- To install Adilib you only need to have installed Docker and Docker-compose. If deployed in a cloud/remote server, you need to have access to the balancer (Apache/NGINX) in order to make Adilib generated Chatbots accessible from outside the server.

Hardware requirements:

These are the basic requirements for a standard deployment with a load of 100 concurrent users:

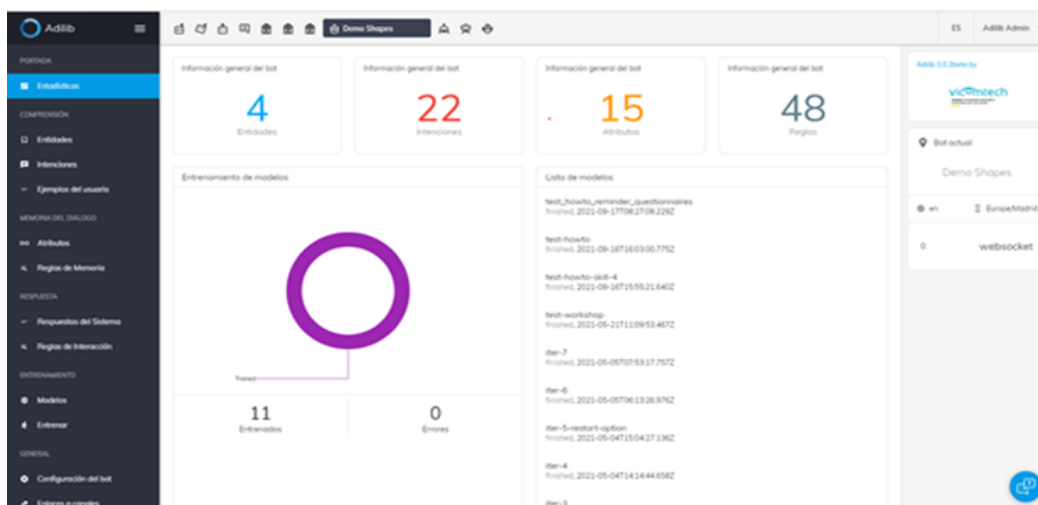
- CPU: Intel(R) Xeon(R) CPU E5-2680 or higher. – IMPORTANT: The CPU must have Advanced Vector Extension (AVX) for proper operation.
- RAM Memory: 16GB or more.
- Hard Disk: 50GB or more. The need will increase depending on the number of trained models, data stored in Elastic Search etc.

1.3.2 Interfaces offered

Two interface-types are provided to use Adilib: web-based interfaces and WebSocket based interfaces. Three main interfaces are available:

Web Interface for building Chatbots

The first interface is a web-based UI which can be used to configure, personalize, train and deploy chatbots. It is intended to be used by technicians with knowledge on chatbot building.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

Figure 21 – Adilib’s web-based interface.

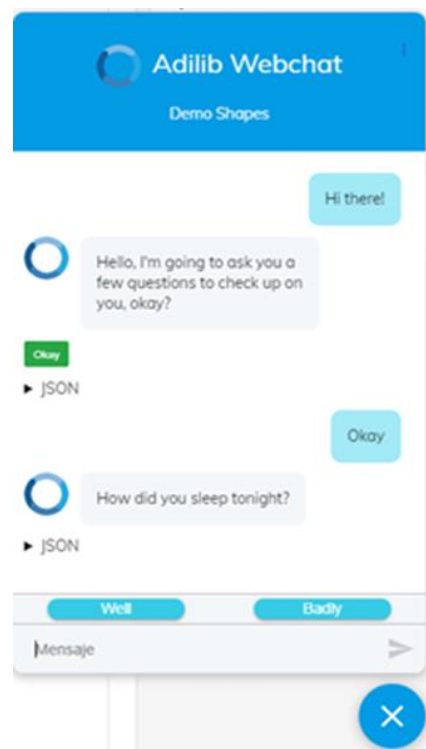


Figure 22 – Webchat widget.

The second interface is a Webchat Widget. It allows to easily add a widget in any modern webpage that can communicate with a chatbot deployed in Adilib.

In addition, if a voice-layer URL is provided it automatically allows a spoken interaction by displaying a microphone-recording button.

The Webchat Widget automatically connects to the WebSocket API of Adilib and renders all the available response-types (images, text, cards, and buttons). In addition, it is highly configurable so it can be adjusted to different use cases.

In order to include the Webchat Widget the following lines of code must be placed inside the <HEAD></HEAD> tags in the main index.html file of the web page:

```
1 <link href="https://cdn.vicomcloud.net/cdn_v2_adilib/css/adil,
  .. ib-widget-dev-1.0.0.css" rel="stylesheet"
  .. type="text/css">
2 <script src="https://cdn.vicomcloud.net/cdn_v2_adilib/js/adil,
  .. ib-widget-dev-1.0.0.js"></script>
```

Then, the following item must be included in a <div> on the webpage to render the widget:

```

1  <div id="webchat"/>
2  <script>
3      WebChat.default.init({
4          debugMode: false,
5          voiceMode: true,
6          selector: '#webchat',
7          socketUrl: '<ws://channel:port>',
8          title: 'Adilib Bot',
9          subtitle: 'Welcome to Adilib',
10         senderPlaceholder: 'Escribe un mensaje...',
11         titleAvatar: 'https://static.thenounproject.com/png/815603-200.png',
12         profileAvatar: 'https://static.thenounproject.com/png/815603-200.png',
13         asr_config: [{ "tag": "es", "label": "Castellano",
14             "voice": "<nombre>", "url": "<voice service url>", "authorization": "Bearer [TOKEN]" },
15             { "tag": "eu", "label": "Euskera", "voice": "<nombre>", "url": "<voice service url>", "authorization": "Bearer [TOKEN]" } ]
16     });
17 </script>

```

With these steps, a widget will be rendered in the desired page, on the bottom right corner, in a closed state.

WebSocket communication API

The WebSocket communication API is used to **consume** the chatbots that are built within Adilib. Once a chatbot is built and trained, a WebSocket **channel** can be created, this channel, deployed in a randomized URI is used to communicate the end users with the chatbot.

The WebSocket communicates using JSON-formatted messages, the Input JSON parameters are specified in the next figure:

Listing 8 JSON Schema of the input message

```

1  {
2      "dialogid": string, // Identifier of the dialogue
3      "message": string, // Message from the user
4      "language": string, // Language identifier (optional)
5      "userid": string, // User identifier (optional)
6      "additionalinfo": {}, // Additional info (optional, interface)
7  }

```

The key to maintaining a conversation with the assistant is to keep the "dialogid" through-out the interactions.

1. In the first interaction send an empty string ("dialogid": "")
2. Adilib will generate a unique identifier for the current dialog/session.
3. In the following calls, send this identifier in the field "dialogid".

To keep the WebSocket open, a simple system of calls must be implemented. Every 30 seconds (approximately), send the string "ping" to the WS. The WS will reply with a "pong" on return, a message to be filtered.

The response given by Adilib is also given in JSON format and consists of different sections, which may be of interest for integration with various systems and front-ends.

Listing 11 Adilib's response structure

```

1  {
2    "dialogid": string, // Dialogue ID
3    "message": [], // Response Messages
4    "language": "eu",
5    "nluinformation": {}, // NLU Response
6    "dminformation": {
7      "semantic-response-list": [], // System response actions
8      "dm-in-out": {
9        "given-output": [], // list of response actions
10       "received-input": {} // Structured NLU input
11     },
12     "dialogue-info": {
13       "dialogue-state-json": {}, // Dialogue State
14       "language": "",
15       "user-id": "" // User ID
16     }
17   },
18   "starttime": "2021-03-03T07:41:09.642452022Z",
19   "endtime": "2021-03-03T07:41:09.731832382Z"
20 }

```

The NLU information structure has the following parameters:

Listing 12 NLU response structure

```

1  {
2      "nluinformation":{
3          "q":{
4              "entities":[
5                  {"confidence": 0.876,
6                   "end": 19,
7                   "entity": "dni",
8                   "extractor": "ner_crf",
9                   "start": 10,
10                  "value": "44334544d"}
11              ],
12              "intent":{ // Intent with greater confidence
13                  "confidence":0.9,
14                  "name":"saludo+__+informar|dni"
15                  // Multiple intents are separated by +__+
16              },
17              "intent_ranking":[
18                  // Ranking of all detected intents
19                  // Same structure as "intent"
20              ],
21              "model":"es_1613664674", // Model used
22              "project":"", // Bot name
23              "text":"Mi DNI es 44334544D" // Input text
24          }
25      },
26  },
27  }

```

The dialogue-state-json has the following parameters:

Listing 13 Dialogue state structure

```

1  {
2      "dialogue-state-json":{
3          "attribute-dict":{
4              "dni": "44334544d"
5          },
6          "system-dict":{
7              "saludar":1
8          },
9          "user-dict":{
10             "informar|dni|44334544d": 1
11         }
12     }
13 }

```

The message list consists of a list of messages of different datatypes, which all of them have the following base structure:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

- Type: defines the data type (text, image, ...)
- Data: Interface that defines the required information to render the message
- Index: Index of the response, i.e., its position in the response

As an example, a plain-text response type JSON is depicted:

Listing 14 Plain text response structure

```

1  {
2      "type": "text",
3      "data": {
4          "value": "Some plain text response"
5      },
6      "index": 0
7  }
```

1.3.3 Test cases and Validation

Adilib has been used in several projects where Vicomtech has the role of chatbot/assistant technology provider, so the whole SDK is being continuously tested in several environments.

In addition, for the SHAPES use case, a Demo Chatbot has been built over the Adilib deployed at <https://adilib.shapes.vicomcloud.net>. This demo bot has been used as a guiding example of the Adilib Workshop and it simulates a Chatbot to gather post-operation information from a user after a knee surgery.

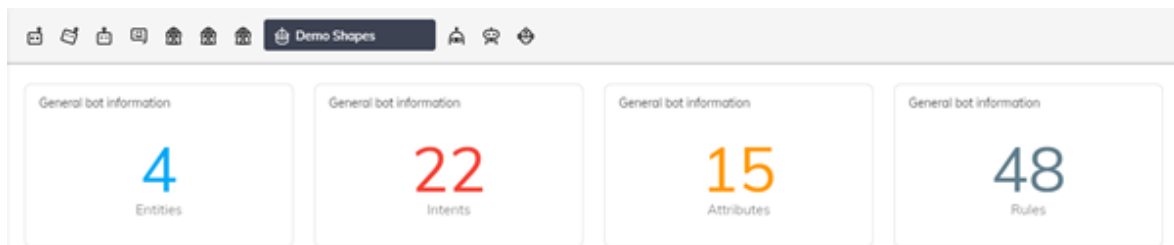


Figure 23 – Demo chatbot.

This demonstrator encompasses a total of 22 intents, 4 entities, 15 memory attributes and 48 interaction rules to build a Chatbot and test the correct integration of all the components of Adilib.

Finally, this chatbot is deployed over a WS channel which is being used by the different pilot-site integrators to test and validate the communication with Adilib.

1.4 At-home Rehabilitation System (UCLM)

UCLM digital solutions are a set of four hardware/software solutions devoted to be tested on real environments with users under pilots “Smart Living Environment for

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

Healthy Ageing at Home”, “Improving In-Home and Community-based Care” and “Physical Rehabilitation at Home”.

These solutions are:

- phyx.io A totem to support rehabilitation
- Activity monitor system: A smart-band devoted to monitor activity of users.
- Fall detection system: a sensor/smart-band combo oriented to detect falls of users.
- Magic mirror component: an assistant service embedded in a mirror/totem devoted to testing interactions with the user.

We would like to point out that UCLM in close connection with SAL is approaching the design and implementation of these solutions under Agile methodology reducing the risk associated to these type of experimental developments.

Up to date, every feature of all solutions have been designed and implemented following user requirements (SAL) in each sprint. We are currently on Validation and Exercises phase 3 and approaching to phase 4 of deployment in Controlled environment according to Gantt defined for the project.

1.4.1 Overview of deployment options

Regarding integration with SHAPES core components, UCLM solutions were chosen to not be integrated with SHAPES core components. Until security scheme is validated and tested for the SHAPES platform, the personal data collected by UCLM digital solutions will keep under end-user control and not external processing, store or analysis will be performed. However, it is expected that a set of trials using fake data will be used to facilitate the integration of UCLM digital solutions in a future when an instance of SHAPES will be deployed compliant with GDPR and security considerations.

1.4.2 Interfaces offered

UCLM digital solutions does not export any API for external services and/or requires data coming from external sources. Only the Call service of the magic mirror requires of Telegram servers.

1.4.3 Test cases and Validation

All the source code generated to test different solutions is under version control in https://bitbucket.org/arco_group/prj.shapes/src/master/ repository. Currently this repository is private, if you need access, please e-mail to Félix Jesús Villanueva (felix.villanueva@uclm.es) to ask access detailing your role inside of SHAPE project and your motivation for getting access to the source code.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 857159.

Phyx.io is a totem devoted to monitor rehabilitation exercises of users previously programmed by the specialists. Each action/role identified in a sprint is validated by the users (SAL) verbally, integrated in the developing and testing pipeline and presented in the following sprint for verbal validation. The features identified till the date according to each role and the test passed are summarized in the following tables.

Currently those activities/roles are tested by 314 unit tests with 94% of coverage of code developed.

| Legend: | | | |
|-------------------|------|-----|-----|
| NOT allowed | | | |
| Allowed | | | |
| FAILS in Backend | BE | BE | BE |
| FAILS in Frontend | FE | FE | FE |
| OK on Backend | BE | BE | BE |
| OK on Frontend | FE | FE | FE |
| Work in Progress | WIP | WIP | WIP |
| Unknown | ?? | ?? | ?? |
| Missing Tests | MT | MT | MT |
| No Support | NS | * | |
| No Need To Test | NNTT | | |

Notes: We distinguish between himself and other users of its same rank
Example: 'Admin: change password'
- column Himself: 'Admin X could change its own password?'
- column Admin: 'Other admins could change Admin X password?'

Figure 24 – Legend used in test monitoring pipeline.

| Action | Anonymous | Himself | Admin | His Manager | A Manager | His Trainer | A Trainer | End User |
|------------------------|-----------|---------|-------|-------------|-----------|-------------|-----------|----------|
| Admin see dashboard | BE-FE | BE-FE | * | * | BE-FE | * | BE-FE | BE-FE |
| list users | BE-FE | BE-FE | * | * | BE-FE | * | BE-FE | BE-FE |
| list facilities | BE-FE | BE-FE | * | * | BE-FE | * | BE-FE | BE-FE |
| list totems | BE-FE | BE-FE | * | * | BE-FE | * | BE-FE | BE-FE |
| list modules | BE-FE | BE-FE | * | * | BE-FE | * | BE-FE | BE-FE |
| list exercises | BE-FE | BE-FE | * | * | BE-FE | * | BE-FE | BE-FE |
| create admin | BE-FE | * | BE-FE | * | BE-FE | * | BE-FE | BE-FE |
| deactivate admin | BE-FE | BE-FE | BE-FE | * | BE-FE | * | BE-FE | BE-FE |
| deactivate user | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| remove admin | BE-FE | BE-FE | BE-FE | * | BE-FE | * | BE-FE | BE-FE |
| remove user | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| change password | BE-FE | BE-FE | BE-FE | * | BE-FE | * | BE-FE | BE-FE |
| edit profile | BE-FE | BE-FE | BE-FE | * | BE-FE | * | BE-FE | BE-FE |
| inactive can not login | * | BE-FE | * | * | * | * | * | * |

Figure 25 – Admin actions developed and tested.

| Action | Anonymous | Himself | Admin | His Manager | A Manager | His Trainer | A Trainer | End User |
|------------------------------|-----------|---------|-------|-------------|-----------|-------------|-----------|----------|
| Manager see dashboard | BE-FE | BE-FE | BE-FE | * | BE-FE | BE-FE | BE-FE | BE-FE |
| see profile | BE-FE | BE-FE | BE-FE | * | BE-FE | BE-FE | BE-FE | BE-FE |
| list users (their/and) | BE-FE | BE-FE | BE-FE | * | BE-FE | * | BE-FE | BE-FE |
| create manager | BE-FE | * | BE-FE | * | BE-FE | * | BE-FE | BE-FE |
| edit profile | BE-FE | BE-FE | BE-FE | * | BE-FE | * | BE-FE | BE-FE |
| remove manager | BE-FE | BE-FE | BE-FE | * | BE-FE | * | BE-FE | BE-FE |
| change password | BE-FE | BE-FE | BE-FE | * | BE-FE | * | BE-FE | BE-FE |
| inactive can not login | * | BE-FE | * | * | * | * | * | * |
| create users | BE-FE | BE-FE | BE-FE | * | BE-FE | * | BE-FE | BE-FE |

Figure 26 – Manager actions developed and tested.

| Action | Anonymous | Himself | Admin | His Manager | A Manager | His Trainer | A Trainer | End User |
|-----------------------------|-----------|---------|-------|-------------|-----------|-------------|-----------|----------|
| Facility see profile | BE-FE | * | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| list facilities | BE-FE | * | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| create facility | BE-FE | * | BE-FE | * | BE-FE | * | BE-FE | BE-FE |
| add totem to facility | BE-FE | * | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| edit information | BE-FE | * | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| remove facility | BE-FE | * | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| Totem see profile | BE-FE | * | BE-FE | BE-FE | BE-FE | * | BE-FE | BE-FE |
| list totems | BE-FE | * | BE-FE | BE-FE | BE-FE | * | BE-FE | BE-FE |
| create totem | BE-FE | * | BE-FE | BE-FE | BE-FE | * | BE-FE | BE-FE |
| edit information | BE-FE | * | BE-FE | BE-FE | BE-FE | * | BE-FE | BE-FE |
| remove totem | BE-FE | * | BE-FE | BE-FE | BE-FE | * | BE-FE | BE-FE |
| show calendar | | | | | | | | |
| Event show event | | | | | | | | |
| add event | | | | | | | | |
| edit event | | | | | | | | |

Figure 27 – Facility, totem and event actions developed and tested.

| Action | Anonymous | Himself | Admin | His Manager | A Manager | His Trainer | A Trainer | End User |
|------------------------|-----------|---------|-------|-------------|-----------|-------------|-----------|----------|
| End User see dashboard | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| see profile | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| create end user | BE-FE | - | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| edit profile | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| change facility | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| deactivate end user | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| remove end user | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| change password | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| inactive can not login | - | BE-FE | - | - | - | - | - | - |
| end user therapists | | | | | | | | |
| end user history | | | | | | | | |
| see profile | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| see inactive profile | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| list users | | | | | | | | |
| | | | | | | | | |

Figure 28 – End-user actions developed and tested.

| Action | Anonymous | Himself | Admin | His Manager | A Manager | His Trainer | A Trainer | End User |
|------------------------|-----------|---------|-------|-------------|-----------|-------------|-----------|----------|
| Trainer see dashboard | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | - | BE-FE | BE-FE |
| see profile | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | - | BE-FE | BE-FE |
| list assigned e. users | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | - | BE-FE | BE-FE |
| create trainer | BE-FE | - | BE-FE | BE-FE | BE-FE | - | BE-FE | BE-FE |
| edit profile | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | - | BE-FE | BE-FE |
| deactivate trainer | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | - | BE-FE | BE-FE |
| remove trainer | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | - | BE-FE | BE-FE |
| change password | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE | - | BE-FE | BE-FE |
| inactive can not login | - | BE-FE | - | - | - | - | - | - |

Figure 29 – Trainer actions developed and tested.

| Action | Anonymous | Himself | Admin | His Manager | A Manager | His Trainer | A Trainer | End User |
|----------|-----------------|---------|-------|-------------|-----------|-------------|-----------|----------|
| Routine | see description | BE-FE | - | BE-FE | - | BE-FE | BE-FE | BE-FE |
| | list | BE-FE | - | BE-FE | - | BE-FE | BE-FE | BE-FE |
| | add routine | BE-FE | - | BE-FE | - | BE-FE | - | BE-FE |
| | edit routine | BE-FE | - | BE-FE | - | BE-FE | BE-FE | BE-FE |
| | run routine | MT | - | MT | - | MT | MT | MT |
| | remove routine | BE-FE | - | BE-FE | - | BE-FE | BE-FE | BE-FE |
| Exercise | see description | BE-FE | - | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| | list | BE-FE | - | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| | add exercise | BE-FE | - | BE-FE | - | BE-FE | - | BE-FE |
| | edit exercise | BE-FE | - | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| | remove exercise | BE-FE | - | BE-FE | BE-FE | BE-FE | BE-FE | BE-FE |
| | | | | | | | | |
| Session | show current | | | | | | | |

Figure 30 – Routine and exercise action implemented and tested.

Phyx.io is django app containerized on Docker but there is no API exported to access data collected by each totem. Currently is designed to run on end-user facilities. A cloud version to manage several facilities is also implemented.

Next steps on Phyx.io testing program is trial of pilot deployment on the three placements of pilots (SAL, CH and DYPE) to be evaluated by SHAPES users not directly involved in Phyx.io developments and sprints. On these trials, users assuming existent roles will be asked to do actions enumerated in previous tables to see if the functionality and GUI is friendly enough or need to be improved. Any bug and/or improve need detected will be included in next sprint as soon as it is detected. After this trial phase the three pilots will be deployed for long-term evaluation and data collection of end-user will be activated. All the partners involved in this pilots will be provided with a link to the issue tracker enabled for Phyx.io in order to report any bug found during pilot lifetime.

Fall detection system is composed by one sensor attached to the end-user and a gateway module with the software which analyses raw data from the sensor. In the gateway, a fall detection machine learning algorithm specifically designed for SHAPES project is running to detect any fall. The performance and architecture has been published and can be consulted in the following reference:

Jesus Fernandez-Bermejo Ruiz, Javier Dorado Chaparro, Cristina Bolanos Peño, Henry A. Llumiguano Solanoa, Xavier del Toro García, Juan C. Lopez Lopez. A low-cost and unobtrusive system for fall detection, 25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems. 2021.

For testing purposes, a dataset was designed. 17 different individuals performed the defined a set of exercises for calibration. One of the issues that were discussed was whether it was appropriate to use a mat to collect the data, as its use may lead to smoothing the acceleration peaks. Otherwise, the individual could perform the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

activities in fear of injury, resulting in data that do not represent a real fall. Finally, the decision was made to collect all data with a mat, as it was considered that it is preferable to have data that approximate a real fall with a slightly smoothed peak. The subjects who performed the activities to create the dataset were 4 females and 13 males, with an average age 30 ± 8.02 years, average height 174.18 ± 7.85 cm, average weight 74.35 ± 9.71 kg. The labelled dataset is available on the website of the ARCO research group: <https://arcoresearch.com/2021/04/16/dataset-for-fall-detection/>.

Next steps on fall detection system testing is to perform trials on pilot emplacements about reliability of the links of sensors with the gateway, the effective communication distance and battery performance. Any bug and/or improve need detected will be included in next sprint as soon as it is detected. After this trial phase the three pilots will be deployed for long-term evaluation and data collection of end-user will be activated. In the pilots, this service will focus on to avoid false positives and to improve performance on real end-users. All the partners involved in these pilots will be provided with a link to the issue tracker enabled for fall-detection service in order to report any bug found during pilot lifetime.

The magic mirror component has two main services, a call service activated with a RFID band and a VICOM component of orofacial rehabilitation. Functional testing of both services has been performed successfully. The trials on pilot scenarios will be devoted to analyse performance on face recognition (light conditions) and audio quality under pilot deployment conditions. All the partners involved in these pilots will be provided with a link to the issue tracker enabled for magic-mirror component in order to report any bug found during pilot lifetime.

The digital solution developed for activity monitoring is based on a smart band connected to a gateway which monitor daily activity. As we mention in fall detection system, next steps in this service is also testing trials on pilot emplacements about reliability of the links of smart band with the gateway, the effective communication distance and battery performance. All the partners involved in these pilots will be provided with a link to the issue tracker enabled for fall-detection service in order to report any bug found during pilot lifetime.

Table 57 – A summary of the testing steps for the UCLM digital solution:

| Digital solution | Test | When | Tool |
|------------------|-----------------------|---------------------------------|--------------------------------------|
| phyx.io | Deployment validation | First month of pilot deployment | Hand-made testing list of procedures |

| | | | |
|-------------------------|--------------------------------|-----------------------------------------|--------------|
| phyx.io | Acceptance of end-user | Periodically on pilot duration | Questionary |
| phyx.io | Acceptance of Trainer | Periodically on pilot duration | Questionary |
| phyx.io | Acceptance of management staff | Periodically on pilot duration | Questionary |
| phyx.io | Bug report | Continuously | Tracker tool |
| Activity monitor system | Acceptance of end-users | First and final month of pilot duration | Questionary |
| Activity monitor system | Bug report | Continuously | Tracker tool |
| Fall detection system | Acceptance of end-users | First and final month of pilot duration | Questionary |
| Fall detection system | Bug report | Continuously | Tracker tool |
| Smart mirror | Acceptance of end-user | Periodically on pilot duration | Questionary |
| Smart mirror | Bug report | Continuously | Tracker tool |

1.5 MedicalSyn Digital Solution (MedSyn)

MedicalSyn offers digital services for medical, clinical-care situations and clinical trials. MedSyn's digital solution an electronic Case report form (eCRF), digital patient surveys, a video consultation tool and graphical tool for medical history. It can be specifically transferred to the needs and requirements of the SHAPES digital solution to connect service providers and improve connection and communication.

1.5.1 Overview of deployment and customization options

In relation to the **Big Data Platform** and the **analytics engine**, it provides outputs following common data models. For each analytical solution the outputs will tend to have a fixed structure. The adaptations of the outputs for each Pilot /Use case will be aligned with the decisions accorded with the partners in charge of visualization of the results and the Pilot/Use case leaders.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857159.

In this sense, the analytic engine is developed to fulfil the main requirements and objectives of the use cases that will be use the results. Therefore, the adaptations of the analytical services for the different use cases are based on the selection of the features that will be displayed within the front-end tools. In any case, the results generated by the analytical engines working within the data platform, meet some requirements:

- The outputs are linked to the IDs of the users.
- The standard output format is *json*.
- Each digital solution provides the outputs following a data model that fulfils the general purposes of the analytic requirements.
- From this data model, the relevant selection of features for each use case will be shown in the front-end services.

In relation to the **Big Data Platform** and the **analytics engine**, the adaptations are linked to the selection of outputs coming from the analytic engines. Following the common requirements of the outputs, the distribution and adaptations of the analytical solutions across the use cases can be summarized for each analytical solution. Three analytical solutions will be working within the big data platform:

1. *Sleep Quality & Physical intensity Level*: it provides the assessment of the sleep quality and the physical activity performed by the participants. The adaptations for the use cases are:
 - PT1-UC-001: visualisations for sleep quality and physical intensity level through eCare. The outputs can feed other analytical solution developed by TREE, Routine and Anomaly detection.
 - PT2-UC-001: visualisations for sleep quality and physical intensity level through eCare complete use of the outputs generated by this analytical engine. Additionally, the outputs will be used as input for recommendation and alert systems developed by VICOM.
 - PT3-UC-001: only a limited selection of outputs is used in this use case like sleep duration and sleep interruptions. For the physical activity assessment only the amount of exercise will be considered ignoring the intensities and kind of activity considerations.
2. *Vitals Control*: based in the calculations of dynamic and personalized threshold for the control of blood glucose, blood pressure, oxygen saturation and heart rate.
 - PT3-UC-general: calculations are based on blood glucose, blood pressure, oxygen saturation and heart rate. Results will be consumed by two different components eCare and eHealthpass.
 - PT3-UC-001: calculations are based only on blood pressure, oxygen saturation and heart rate. Results will be consumed only by eCare.
3. *Routine and Anomaly detection*: analytical solution developed to detect anomalies in the normal sequences of activities (routines) from a smart-home sensor network.
 - PT1-UC-001: The analytical solution is fully dedicated to this use case.

Annex 2 Integration of Digital Solutions with ASAPA and Front-end App.

ASAPA, being a technology- and framework- agnostic authentication scheme, is used by a wide variety of Digital Solutions in the SHAPES project that make use of different frameworks, programming languages, and underlying infrastructure. By offering a generic RESTful API, ASAPA can be easily integrated into any technology a Digital Solution is built upon, making it a robust and self-explanatory Authentication mechanism.

2.1 *Conversion of needs to specifications for integration*

Digital Solution providers contributed to the development of needs so that the API of the authentication component can support easy the integration of these solutions, without having the burden of making technology-specific customizations, and similar trade-offs.

In addition, the usage of an authentication token, removes the responsibility of storing sensitive personal data in the Digital Solutions' database. The Digital Solution sends the authentication credentials (user email and password), and receives an authentication token which can be then used in all requests towards the SHAPES framework. Even if this token is eavesdropped, it neither contains no information about the user, nor can it be used past its expiration date (set also by the ASAPA scheme).

2.2 *Refactoring of a Digital Solution to accommodate ASAPA*

Most Digital Solutions already offer a dedicated Authentication Scheme. In most cases, a Digital Solution might connect to an external or internal database, in order to perform authentication, and then fetch and store user-specific data.

With ASAPA, an additional layer of authentication should be added. The Digital Solution needs to extend its own authentication process, in order to be able to make a REST call to the ASAPA as well, then, the Digital Solution stores the ASAPA authentication token and is ready to use it when making REST calls to the SHAPES component ecosystem.

2.3 *Integrating Front-end App with a Digital Solution*

If a Digital Solution is already installed on the same Android handheld device as the Front-End-App, and the Front-End app includes the necessary information to open the Digital Solution, then the Digital Solution is not required to implement the ASAPA authentication component to authenticate the user. The Front-end app can perform the authentication with ASAPA, and then trigger each Digital Solution to open, by also

passing the authentication token. Then, the Digital Solution also stores this token and uses it in any request towards SHAPES infrastructure.

If the user opens a Digital Solution directly (without opening the Front-end app first), the Digital Solution should then trigger the Front-end app in order to perform the authentication.

Additionally, if the authentication token expires and the authentication with ASAPA should be performed again, the Digital Solution should be able to trigger the Front-end app again in order to perform the authentication, and return to the Digital Solution with new authentication.