Deliverable D3.3

MATISSE prototype testing evaluation

MATiSSE

Empower Hospitals' Reliability via TagItSmart Service platform

Funded by TagItSmart - Smart Tags driven service platform for enabling ecosystems of connected objects
Information and Communication Technologies (ICT) Programme
H2020-ICT-30-2015
Project No: 688061
Start date of the project: 1 January 2016
Duration: 36 months

D3.3 – version 1.0 – 25/05/2018

PROGRAMME NAME:	H2020-ICT-2015
PROJECT NUMBER:	688061
PROJECT TITLE:	TagItSmart/MATISSE
COORDINATOR:	Therapaenis P.C. [THERA] (EL)
DOCUMENT NUMBER:	D3.3/THERA/WP3/v1.0
WORK-PACKAGE:	WP3
DELIVERABLE TYPE:	Report
CONTRACTUAL DATE OF DELIVERY:	30/04/2018
ACTUAL DATE OF DELIVERY:	25/05/2018
TITLE OF DOCUMENT:	MATISSE prototype testing evaluation
AUTHOR(S):	Terpsi Velivassaki (THERA)
APPROVAL OF THIS REPORT:	Chris Zachariadis (THERA)
SUMMARY OF THIS REPORT:	See the Executive Summary
HISTORY:	See the Change History Table
KEYWORD LIST:	MATISSE, prototype, smart cart, test, validation
AVAILABILITY:	Public (PU)

Disclaimer

This document may contain material that is copyright of Therapaenis, and may not be reproduced or copied without permission. Therapaenis agrees to the full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

Document Revision History

Date	Version	Author	Summary of main changes
02/03/2018	0.1	T. Velivassaki, THERA	Table of Contents
26/04/2018	0.2	T. Velivassaki, THERA	Initial input for most of the tests.
04/05/2018	0.4	T. Velivassaki, THERA	Added the results of the Smart Cart Testing.
11/04/2018	0.5	Ch. Zahariadis, THERA	First peer review
16/05/2018	0.6	T. Velivassaki, THERA	Fixes to address comments from the first peer review. Added test code examples in §4.1 and Annex A.
21/05/2018	0.8	T. Velivassaki, THERA	Added Executive Summary, Introduction and Conclusions
23/05/2018	0.9	Ch. Zahariadis, THERA	Quality Check and second peer review
25/05/2018	1.0	T. Velivassaki, THERA	Final Version

Executive Summary

This deliverable targets at setting the baseline of features and functional characteristics against which the satisfaction of the MATISSE requirements already identified in deliverable D2.1 [1] will be validated. The relevant procedure is formulated with the help of concrete and structured test case templates guaranteeing that, when successfully implemented, MATISSE will be compliant with the specifications defined in deliverable D2.1.

Under the perspective of this structured, requirements satisfaction-oriented testing approach, section 2 provides an overview of the high-level testbed architecture which matches the envisioned MATISSE architecture. In section 3 the adopted testing methodology is presented, followed by the detailed test reporting form (template). In section 4, the testing procedures and the respective results are documented whereas section 5 concludes the document. Finally, in Annex A, the source code for the combined MATISSE Drugs' Stocktaking and In-Hospital Medication use cases is provided, for reference.

Table of Contents

1. Introduction 9
1.1. Scope of the deliverable 9
1.2. How to read this document 9
2. MATISSE Testbed 9
3. MATISSE Testing Methodology 10
3.1. Methodology 10
3.2. Test Reporting Form 11
4. Tests and Results 12
4.1. MATISSE Back-End Testing 12
4.1.1 Test item 12
4.1.2 Objective 12
4.1.3 Features to be tested 12
4.1.4 Test Prerequisites 13
4.1.5 Test Description 13
4.1.6 Input 16
4.1.7 Test Setup 17
4.1.8 Expected Result 17
4.1.9 Outcome 18
4.1.10 Checkpoints 19
4.2. MATISSE Dashboard Testing 19
4.2.1 Test Item 20
4.2.2 Objective 20
4.2.3 Features to be Tested 20
4.2.4 Test Prerequisites 20
4.2.5 Test Description 20
4.2.6 Test Setup 21
4.2.7 Expected Result 21
4.2.8 Outcome 21
4.2.9 Checkpoints 26

4.3. MATISSE Smart Cart Testing 26
4.3.1 Test Item 27
4.3.2 Objective 27
4.3.3 Features to be Tested 27
4.3.4 Test Prerequisites 27
4.3.5 Test Description 27
4.3.6 Input 28
4.3.7 Test Setup 28
4.3.8 Expected Result 29
4.3.9 Outcome 29
4.3.10 Checkpoints 32
4.4. MATISSE Mobile Application Testing 33
4.4.1 Drugs' Stocktaking 33
4.4.2 In-Hospital Medication 46
4.4.3 Smart Medical Exams 52
4.4.4 Pre-surgery Check 55
4.4.5 Patients' Monitoring 58
5. Conclusion 63
6. References 63
Annex A 64

List of Figures

Figure 1: The MATISSE testbed simplified deployment architecture. 9
Figure 2: The complete list of API categories exposed by the MATISSE Back-End component. 13
Figure 3: The set of RESTful API service endpoints exposed by the allergies API category. 14
Figure 4: An example of the tests conducted on each endpoint. 15
Figure 5: Initialization of the test database 17
Figure 6: The user enters their credentials 22
Figure 7: The signed in user visualizes a data table with Allergy records 22
Figure 8: The patient fills the form to register a new allergy 23
Figure 9: It is confirmed that the data table is refreshed after the addition of the new record 23
Figure 10: The available options that can be performed on a record 24

Figure 11: The user alters the “Type of Allergy” field 24
Figure 12: The record has been updated 25
Figure 13: The last record has been deleted 25
Figure 14: A welcome message appears on the screen 30
Figure 15: The tester’s and patient’s names appear on the LCD display and the requested bin’s drawer LED is on 30
Figure 16: The requested bin’s LED is on 31
Figure 17: The LED is switched off upon request 31
Figure 18: The Smart Cart is shut down after requested 32
Figure 19: Signing in the MATISSE Mobile Application 37
Figure 20: The main menu of the mobile application (a). The scanner is automatically activated and the Smart Cart’s tag is scanned (b) 38
Figure 21: The applications has fetched the first patient with his prescriptions’ doses 39
Figure 22: The patient is fetched and the corresponding drawer’s LED in on. The patient’s and tester’s name appear on the LCD display 40
Figure 23: The tester opens the drawer and the corresponding patient’s bin LED is on 40
Figure 24: The drugs’ packagings (thngs) with the initial values of “Number of Pills” properties .. 41
Figure 25: The tester selects Drug 2 and scans it. It is the correct so the dose is removed from the list. 42
Figure 26: The tester should have scanned Drug 3 but, mistakenly, scans Drug 5. The application raises a related alert. 43
Figure 27: The values of “Number of Pills” property for the corresponding drugs are reduced 44
Figure 28: The current patient has no more prescribed drugs, the tester selects “NEXT PATIENT” and the next patient is fetched 45
Figure 29: The QR scanner launched automatically when the In-Hospital Medication option was selected. 49
Figure 30: The application fetches the scanned patient and an overview of the cart. It is confirmed the drawer from the app and the drawer with the LED on are the same. 50
Figure 31: The tester selected the drawer from the app’s screen, opened the correct drawer on the smart cart and confirms that the bin denoted from the app is the same as the one with the LED on, on the cart 50
Figure 32: The tester selects the bin through the app, and the corresponding LED on the Smart Cart switches off 51
Figure 33: The tester selects “SCAN NEXT PATIENT”, scans another patient and the application fetches the data as in the previous patient. 51
Figure 34: The patient’s tag is scanned (a) and his exam record is visualized (b) on the screen 54
Figure 35: The tester visualizes specific exams 55
Figure 36: The QR scanner is launched for Pre-surgery Check 57
Figure 37: The pending surgery appears and the tester visualizes its details 58
Figure 38: The QR scanner launches automatically for Patient’s Monitoring 60
Figure 39: The testers enters the Patient’s Monitoring session and visualizes a list of exams 61
HYPERLINK "bookmark://Figure!37|sequence" ..... OBJ

List of Abbreviations

AM	Access Manager
API	Application Program Interface
CRUD	Create, Read, Update, Delete
DS	Data Storage
EC	European Commission
EM	Entity Manager
FC	Functional Code
GPIO	General Purpose Input/Output
HTTP	Hypertext Transfer Protocol
LED	Light-emitting diode
PQM	Process Quality Manager
QR	Quick Response
REST	Representational State Transfer
RPi	Raspberry Pi
SM	Session Manager
SCM	Smart Cart Manager
SSc	SmartTag Scanner
TiS	TagItSmart!
TTI	Time-Temperature Indicator
UI	User Interface
UM	User Manager
WP	Work Package

1. Introduction

1.1. Scope of the deliverable

The present document constitutes the third deliverable of the Work Package (WP) 3 of the MATISSE project, handling the MATISSE prototype testing evaluation and essentially reflecting the work performed in the context of Task 3.3 namely “System integration and testing”.

The scope of this deliverable is to document the testing procedures that were followed throughout the functional evaluation of MATISSE, to guarantee that the requirements identified and documented in deliverable D2.1 are satisfied by the final prototype implemented.

1.2. How to read this document

Since this deliverable targets at simply documenting the evaluation procedure and the relevant outcomes, the internal operation of the components or their interworking are not covered. A quick overview of the MATISSE testbed, matching the intended architecture is presented in the next paragraph, as an introduction. For more information, the interested reader should refer to deliverables D2.1, D3.1 [2], D3.2 [3] and D4.1 [4].

Assuming that the reader is already familiar with the MATISSE scope, architecture and requirements, this document is meant to be read in a linear manner.

2. MATISSE Testbed

As reported in deliverable D3.1 [2], the first integrated prototype of the MATISSE platform is composed of the following components:

- The initial version of the MATISSE Back-end;
- The initial version of the MATISSE Dashboard;
- The initial version of the MATISSE Mobile Application;
- The initial version of the MATISSE Smart Cart.

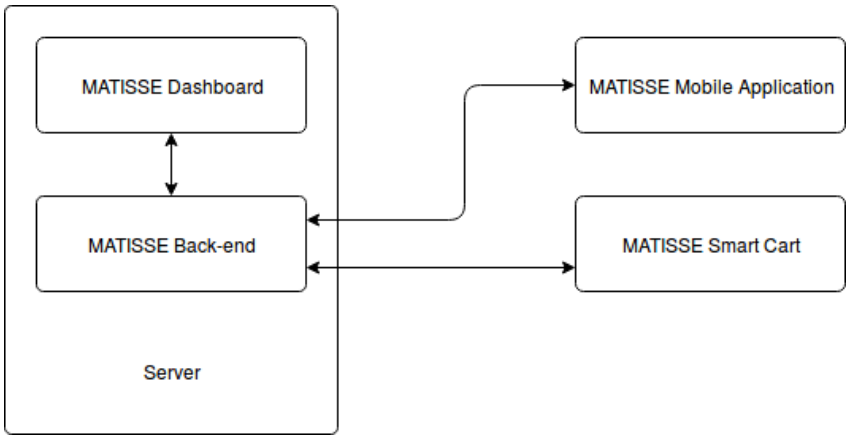


Figure 1: The MATISSE testbed simplified deployment architecture.

As depicted in Figure 1, the first integrated prototype of the MATISSE platform is composed of four entities:

- The MATISSE Back-end which lies on the server and implements five components of the proposed MATISSE system architecture, the Smart Cart manager (SCM), responsible for the management and control of the smart cart's operation, the Data Storage (DS) which supports the storage of all the necessary information for the operation of the MATISSE platform, the TiS Handler, responsible for integrating TiS functionalities with the MATISSE platform, the Entity Manager (EM) which caters for the processing of information related to any kind of Entity appearing in the MATISSE platform and the Process Quality Manager (PQM) which keeps track of the activities performed within any of the MATISSE components. The MATISSE Back-end communicates with all the other components of the platform.
- The MATISSE Dashboard, which also lies on the server and implements two architectural components of the proposed architecture, the Access Manager (AM) which supports the authentication and authorization services and the User Manager (UM) which is responsible for managing the users of the MATISSE platform. The MATISSE Dashboard communicates only with the MATISSE Back-end and provides a Graphical User Interface (GUI) for the users of the MATISSE platform.
- The MATISSE Mobile Application, which implements three architectural components of the platform, the Access Manager (AM), the Session Manager (SM) which concerns the Smart Cart related use cases, and the Smart tag Scanner (SSc) which enables the scanning of smart tags. This component directly communicates only with the back-end. It also communicates with the Smart Cart via the back-end.
- the MATISSE Smart Cart which is a hardware component and, along with the MATISSE Mobile application, supports the execution of the two main use cases of the MATISSE platform, Drugs' Stocktaking and In-Hospital Medication. The Smart Cart directly communicates with the MATISSE Back-end component.

3. MATISSE Testing Methodology

3.1. Methodology

The MATISSE Testing Methodology dictates that each test to be executed should include the following steps:

- Identify objectives, which create the need for conducting the test, driven by the MATISSE functional requirements;
- Identify the test items, i.e. the software and hardware components that need to be tested against the aforementioned objectives;
- Identify features of these components to be tested;

- Specify the test approach depending on the role of each component under testing, and whether it can be tested as standalone or in combination with another component;
- Specify any requirements posed by the identified components, in order for them to be operational;
- Define the hardware deployment of the aforementioned components;
- Execute the test;
- Observe the outcome;
- Report the test conditions and results.

Section 4 is split into multiple paragraphs, each one documenting the identified tests as well as their execution outcome, as detailed above.

3.2. Test Reporting Form

Test documentation is important, as it allows future access to test data and results, enabling higher exploitation or even reuse of the results by the same or third-party researchers, given that access to them is allowed. The documented information should at least describe the conditions under which the tests have been designed and executed, reflecting the test methodology, as well as observations during test execution and the results. The main objective of testing is to verify the functionalities designed for MATISSE are well supported in the current prototype.

Test Item *Description of the item under test* **Objective** *Description of the test objectives* **Feature(s) to be tested** *The features under testing for each test item* **Test Prerequisites** *A list of required actions to ensure the successful test execution* **Test Description** *A list of actions composing the test*

Input

The input data used for the test (if existent)

Test Setup

The MATISSE components involved in the test

Expected Result

The expected outcome of the test, based on reference data and/or executions

Outcome

The actual result of the conducted test

Checkpoints

A list of points to be checked during the test execution

4. Tests and Results

4.1. MATISSE Back-End Testing

The MATISSE Back-End component holds a significant role in the operation of MATISSE, being responsible for serving all data requests issued by the MATISSE Mobile Application and the Dashboard, simultaneously coordinating with the Smart Cart component and controlling its operation. In this framework, to ensure the validity of the Back-End component, the relevant testing suite includes unit tests for all the services exposed by the component. For the purposes of testing, the APIClient offered by Django REST Framework [5] (the technology used for implementing the web services part of the Back-End component) has been utilized.

4.1.1 Test item

The item under test is the MATISSE Back-End component of MATISSE.

4.1.2 Objective

The test objective is to verify the corect operation of the implemented RESTful API in terms of response codes of the issued requests, as well as its contextual operation, the semantic validity of the actual web services responses with respect to the expected ones.

4.1.3 Features to be tested

The following features are going to be tested:

- The response codes for every kind of request
- The semantic correctness of the implemented REST API calls

The two identified features, though phenomenally disjoint, are tightly integrated, since according to IETF RFC 7231 [6], the response codes of HTTP requests should be tightly bound to their contextual and semantic state, e.g. if a user is not authenticated against the system the response code should be 401.

4.1.4 Test Prerequisites

The Data Storage (DS) component is deployed and configured.

4.1.5 Test Description

The set of API categories exposed by the MATISSE Back-End component as visualized by the Swagger [7] dynamic documentation system is presented in Figure 2.

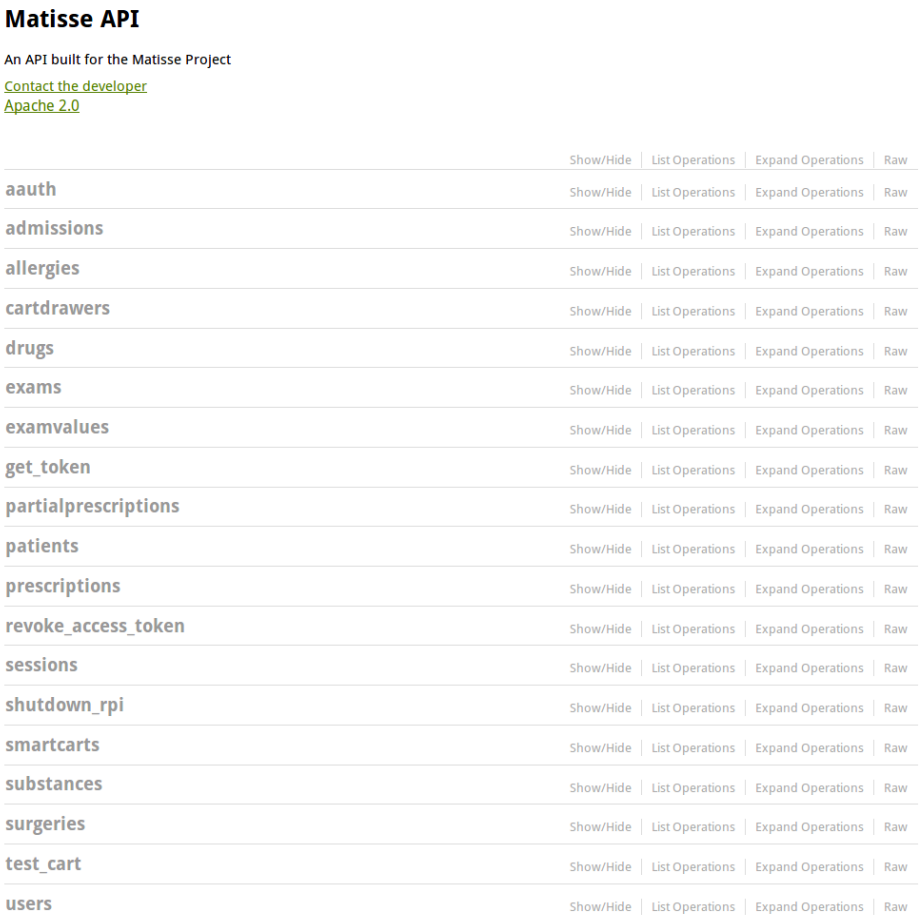


Figure 2: The complete list of API categories exposed by the MATISSE Back-End component.

Since each API category consists of a set of RESTful API endpoints exposing distinct functionalities to be used either by the MATISSE Dashboard, the MATISSE mobile application and the MATISSE smart cart (see Figure 3 for an indicative case), different tests for the entirety of API service endpoints have been introduced.

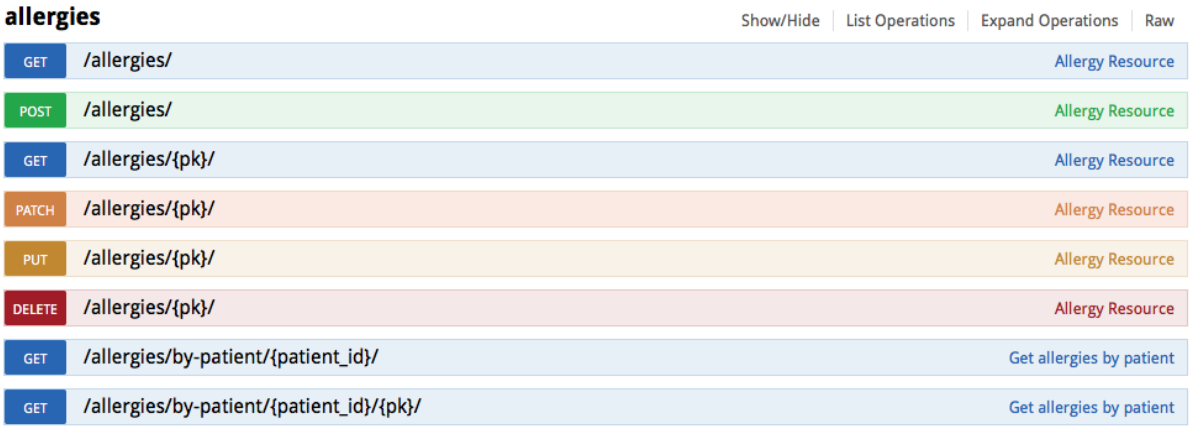


Figure 3: The set of RESTful API service endpoints exposed by the allergies API category.

First, the testing of every implemented REST API call in terms of HTTP response codes is tested. Figure 4 delivers an example of the relevant test for the RESTful API service endpoints pertaining to the API category of allergies.

1. The tester issues requests with no credentials to every API service endpoint and confirms that 401 response code is returned.
2. The tester issues requests with access tokens of specific scopes and checks that operations on API service endpoints out of scope return a 403 response code.
3. The tester issues requests on non-existent objects of every API service endpoint type and confirms that a 404 response code is returned.
4. The tester issues requests with a wrong format on every API service endpoint type and confirms that a 400 response code is returned.
5. The tester issues requests of not supported method types and confirms that a 405 response code is returned.
6. The tester issues GET, PUT, PATCH requests, proper for every API service endpoint and confirms that a 200 response code is returned.
7. The tester issues POST requests, proper for every API service endpoint, and confirms that a 201 response code is returned.
8. The tester issues DELETE requests on an object of all API service endpoint types and confirms that a 204 response code is returned.

```
self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' + self.token_psurgery.token)
self.assertEqual(self.test_client.get('/allergies/').status_code, 403)
self.assertEqual(self.test_client.post('/allergies/', self.allergy_data, format='json').status_code, 403)
self.assertEqual(self.test_client.get(self.allergy_2_url).status_code, 403)
self.assertEqual(self.test_client.delete(self.allergy_2_url).status_code, 403)
self.assertEqual(self.test_client.patch(self.allergy_2_url).status_code, 403)
self.assertEqual(self.test_client.put(self.allergy_2_url, self.allergy_data, format='json').status_code, 403)
self.assertEqual(self.test_client.get('/allergies/by-patient/' + str(self.patient_1.id) + '/').status_code, 403)
self.assertEqual(self.test_client.get('/allergies/by-patient/' + str(self.patient_1.id) + '/' + str(self.allergy_1.id) + '/').status_code, 403)

self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' + self.token_clinician.token)
self.assertEqual(self.test_client.get('/allergies/').status_code, 200)
self.assertEqual(self.test_client.post('/allergies/', self.allergy_data, format='json').status_code, 201)
self.assertEqual(self.test_client.get('/allergies/' + str(self.allergy_2.id + 1) + '/').status_code, 200)
self.assertEqual(self.test_client.patch('/allergies/' + str(self.allergy_2.id + 1) + '/').status_code, 200)
self.assertEqual(self.test_client.put('/allergies/' + str(self.allergy_2.id + 1) + '/', self.allergy_data, format='json').status_code, 200)
self.assertEqual(self.test_client.delete('/allergies/' + str(self.allergy_2.id + 1) + '/').status_code, 204)
self.assertEqual(self.test_client.get('/allergies/by-patient/' + str(self.patient_1.id) + '/').status_code, 200)
self.assertEqual(self.test_client.get('/allergies/by-patient/' + str(self.patient_1.id) + '/' + str(self.allergy_1.id) + '/').status_code, 200)

self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' + self.token_super.token)
self.assertEqual(self.test_client.get('/allergies/').status_code, 200)
self.assertEqual(self.test_client.post('/allergies/', self.allergy_data, format='json').status_code, 201)
self.assertEqual(self.test_client.get('/allergies/' + str(self.allergy_2.id + 2) + '/').status_code, 200)
self.assertEqual(self.test_client.patch('/allergies/' + str(self.allergy_2.id + 2) + '/').status_code, 200)
self.assertEqual(self.test_client.put('/allergies/' + str(self.allergy_2.id + 2) + '/', self.allergy_data, format='json').status_code, 200)
self.assertEqual(self.test_client.delete('/allergies/' + str(self.allergy_2.id + 2) + '/').status_code, 204)
self.assertEqual(self.test_client.get('/allergies/by-patient/' + str(self.patient_1.id) + '/').status_code, 200)
self.assertEqual(self.test_client.get('/allergies/by-patient/' + str(self.patient_1.id) + '/' + str(self.allergy_1.id) + '/').status_code, 200)

# Tests for Bad Request, Not Found, Method Not Allowed (400, 404, 405)
self.assertEqual(self.test_client.post('/allergies/', {}).status_code, 400)
self.assertEqual(self.test_client.post('/allergies/', {'patient': self.patient_1_url}, format='json').status_code, 400)
self.assertEqual(self.test_client.put('/allergies/' + str(self.allergy_1.id) + '/', {'patient': '/patients/3/'}, format='json').status_code, 400)
self.assertEqual(self.test_client.get('/allergies/by-patient/one/').status_code, 400)

self.assertEqual(self.test_client.get('/allergies/25/').status_code, 404)
self.assertEqual(self.test_client.patch('/allergies/25/').status_code, 404)
self.assertEqual(self.test_client.put('/allergies/25/', self.allergy_data, format='json').status_code, 404)
```

Figure 4: An example of the tests conducted on each endpoint.

Having concluded on the API correctness at basic operation and response codes level, the automated testing of the MATISSE Back-End component was continued with the semantic, use-case based variant (see deliverable D4.1 [4] for details and discussion related to use cases definition), successively including:

1. The Drugs' Stocktaking use case;
2. In-Hospital Medication use case.

The testing of the Drugs' Stocktaking use case comprises the following steps:

1. The Administrator is set as the current user.
2. Prescriptions are created for patients, with doses that need to be received shortly after the creation.
3. The Prescription Handler is set as the current user.
4. A request is issued to the endpoint which is responsible for associating the patients that have pending drugs with the Smart Cart drawers.

5. We ensure that the patients are correctly associated uniquely with every drawer.
6. A request is issued and the doses that each patient should receive during this session are fetched.
7. Every dose of a medicine is marked as ready and the prescription handler is marked as the current user through another request.
8. We ensure that the prescription doses are marked as ready and their handler is set as the current user.

Having validated the MATISSE contextual validity against the Drugs' Stocktaking use case, we tested against the In-Hospital Medication use case, which includes the following steps:

1. The Medication Distributor is set as the current user.
2. Supposedly, the patient is scanned, and its corresponding drawer is fetched.
3. The drugs are distributed and a request is issued to mark the doses as completed and set the distributor to the current user.
4. We ensure that the prescription doses are marked as completed and their distributor is set as the current user.
5. We ensure that the patients are disassociated from the Smart Cart bins.

The aforementioned steps conclude the automated testing of the MATISSE Back-End component. As an indication of how the test was actually performed, the relevant source code test file is provided in Annex A.

4.1.6 Input

The first step for the testing is the preparation of the test data that will be employed for the evaluation of the REST API operation. To begin with, a testing client is initialized and a testing application is added to this client. Afterwards, sample objects are created for each and every API service endpoint available by the REST API, as well as serialized data for the testing of POST requests.

It should be mentioned that the MATISSE Back-End supports 6 different types of users, namely:

1. the Administrator;
2. the Prescription Handler;
3. the Medication Distributor;
4. the Medical Exams Handler;
5. the Surgery Preparation Handler;
6. The Clinician.

One user for each type is created in the testing application. Additionally, in order to evaluate our implementation of the OAuth2.0 protocol, tokens of different scopes are created for each user. In Figure 5 some of the initialization is presented.

```
class APITestCases(TestCase):

    @classmethod
    def setUpTestData(cls):
        """ Generating Test Data """
        cls.test_client = APIClient(ACCEPT='application/json')
        cls.now = timezone.now()
        cls.today = cls.now.date()

        # Create users for Testing
        cls.user_super = User.objects.create_user(username='admin', password='admin12345', is_superuser=True)
        cls.user_1_url = '/users/' + str(cls.user_super.id) + '/'

        cls.user_phandler = User.objects.create_user(username='phandler', password='phandler12345')
        cls.user_phandler.profile.user_type = 'PrescriptionHandler'
        cls.user_phandler.profile.save()
        cls.user_phandler.save()
        cls.user_2_url = '/users/' + str(cls.user_phandler.id) + '/'

        cls.user_meddist = User.objects.create_user(username='meddist', password='meddist12345')
        cls.user_meddist.profile.user_type = 'MedicationDistributor'
        cls.user_meddist.profile.save()
        cls.user_meddist.save()
        cls.user_3_url = '/users/' + str(cls.user_meddist.id) + '/'

        cls.user_medexams = User.objects.create_user(username='medexams', password='medexams12345')
        cls.user_medexams.profile.user_type = 'MedicalExamsHandler'
        cls.user_medexams.profile.save()
        cls.user_medexams.save()
        cls.user_4_url = '/users/' + str(cls.user_medexams.id) + '/'

        cls.user_psurgery = User.objects.create_user(username='psurgery', password='psurgery12345')
        cls.user_psurgery.profile.user_type = 'SurgeryPreparationHandler'
        cls.user_psurgery.profile.save()
        cls.user_psurgery.save()
        cls.user_5_url = '/users/' + str(cls.user_psurgery.id) + '/'
```

Figure 5: Initialization of the test database

4.1.7 Test Setup

The MATISSE components involved in this test are the following:

- Data Storage (DS)
- Smart Cart Manager (SCM)

4.1.8 Expected Result

The unit tests that we initially developed were intended for the testing of the Create, Read, Update, Delete (CRUD) HTTP requests on the RESTful API service endpoints by the their response codes. The following client error and success response codes are expected:

- **401 Unauthorized**: Expected when an anonymous request is issued on a protected API service endpoint.
- **403 Forbidden**: Expected when access to a specific scope is attempted with an out-of-scope access token.
- **400 Bad Request**: Expected when a POST operation with invalid data is issued to a API service endpoint.
- **404 Not Found**: Expected when a request on an object that does not exist might have been performed.
- **405 Method Not Allowed**: Expected when the requested method is not allowed on the specific API service endpoint.
- **200 OK**: Expected when an appropriate operation on the requested API service endpoint has been processed and is successful.
- **201 Created**: Expected when an appropriate POST request has been issued and the requested API service endpoint has been successfully created.
- **204 No Content**: Expected when a DELETE request has been issued and the requested API service endpoint has been successfully deleted.

After the successful testing of CRUD operations on each and every API service endpoint, and for every type of user, the testing of use case scenarios follows. The initial testing solely involves the MATISSE Back-End component, even during the Smart Cart based scenarios. The use cases that are tested are Drugs' Stocktaking and In-Hospital Medication.

By the end of the back-end testing of the Drugs' Stocktaking use case each of the patients with pending prescriptions should have been associated with a specific Smart Cart bin and their prescription doses should have been marked as ready.

By the end of the back-end testing of the In-Hospital Medication use case each of the prescription doses should have been marked as completed and all the patients should have been disassociated from their corresponding Smart Cart bins.

4.1.9 Outcome

The automated testing process has been conducted repeatedly, both during and after the development of the MATISSE Back-End component. The testing has been successful as displayed below.

```
$ python3 manage.py test –settings=matisse.settings.test
Creating test database for alias 'default'…
System check identified no issues (0 silenced).
................
----------------------------------------------------------------------
Ran 16 tests in 18.252s

OK
Destroying test database for alias 'default'...
```

4.1.10 Checkpoints

Checkpoint	Result	Comments
Is 401 Unauthorized returned on unauthorized requests?	PASS	All kinds of requests were issued to every available endpoint without any credentials.
Is 403 Forbidden returned when a type of user cannot access a specific API service endpoint?	PASS	Besides the Administrator, not every type of user is authorized to have access on specific API service endpoints.
Is 404 Not Found returned when a specific object does not exist?	PASS	Requests were attempted on non-existent objects in order to ensure that they are not found.
Is 400 Bad Request returned when a request with wrong format is issued?	PASS	Requests with wrong URL parameters, invalid format in JSON Objects etc. were issued to every endpoint.
Is 405 Method Not Allowed returned when a non-available method is requested?	PASS	POST requests were issued to endpoints that support only GET operations etc.
Is 200 OK returned when an operation has been successful?	PASS	Upon GET, PUT and PATCH operations.
Is 201 Created returned when a new object has been created?	PASS	A POST request is issued for the creation of a new object on a API service endpoint.
Is 204 No Content returned when the deletion of an object is attempted?	PASS	A DELETE request is issued on an object.
Are patients associated with bins and have ready prescription doses at the end of the Drugs' Stocktaking session?	PASS	Patients with pending prescriptions are associated with bins and their prescriptions are marked as ready.
Are patients disassociated with bins and have completed prescriptions at the end of the In-Hospital Medication session?	PASS	Patients are not associated with bins anymore and their prescriptions are marked as completed.

4.2. MATISSE Dashboard Testing

The dashboard serves as a bridge of communication between the end-user and the MATISSE Back-End. Through this component, the user can retrieve, add, update or delete records. The secretariat can use the dashboard to register new patients, keep track of their admissions, scheduled or completed surgeries, allergies, exams, active and past prescriptions, while a clinician may use this component to write a new prescription for a patient or obtain an overview of their medical history.

The dashboard also offers an overview of the drugs that are used in the hospital, along with their active substances.

4.2.1 Test Item

The item under test is the MATISSE Dashboard component of MATISSE.

4.2.2 Objective

The test objective is to verify the capabilities of creating, reading, updating and deleting an object as an end-user.

4.2.3 Features to be Tested

The following features are going to be tested:

- Creating new objects through the available forms
- Reading objects through the available data tables
- Updating Objects
- Deleting Objects

4.2.4 Test Prerequisites

- The MATISSE Back-End component is deployed and configured
- There is at least one registered user as Administrator

4.2.5 Test Description

1. The tester accesses the MATISSE Dashboard's login page and signs in.
2. The tester navigates to the desired resources through the side panel of the dashboard and visualizes the stored records in data tables.
3. The tester selects the "Add" button of a resource and a form appears.
4. The tester fills the required data in the form and clicks "Submit".
5. The tester visualizes the updated data table for this resource.
6. The tester selects a record to update and a form filled with data out of this record appears.
7. The tester alters the fields he desires and clicks "Submit".
8. The tester visualizes the updated data for this resource.
9. The tester selects a record to delete.
10. The tester visualizes the updated data for this resource.

11. The tester signs out.

4.2.6 Test Setup

The MATISSE components involved in this test are the following:

- Data Storage (DS)
- Access Manager (AM)
- User Manager (UM)

4.2.7 Expected Result

The tester, after signing in the MATISSE Dashboard navigates via the sidebar and should be able to visualize the data that already exists in the Data Storage. The tester should then be able to select the addition of a new record for any resource and, after filling in a form, he/she should be able to visualize the updated data.

Additionally, the tester should be able to edit and update a specific record by selecting “Edit” from the available actions. The form with the data should appear and after editing, the “Submit” button is pressed, and the tester visualizes the updated data once again. Moreover, the tester should be able to select “Delete” from the available actions and the record should not appear in the data table anymore.

4.2.8 Outcome

The test results are presented as snapshots from the MATISSE Dashboard component, displaying the results of tester’s actions. The following process is repeated for every resource that can be accessed through the dashboard and the results were the expected ones.

Firstly, the login page appears, as shown in Figure 6 and the tester is not able to visualize any records of the available resources if he is not signed in. So, he/she enters his/hers credentials and clicks “Sign In”. Then he/she is able to navigate through the sidebar to any of the available resources. In Figure 7, a data table of recorded allergies is shown.

By clicking on the “+” symbol on the panel’s top right corner a form with the fields required by the Allergy resource shows up and the user fills the form as shown in Figure 8. After submitting the form, the updated Allergy data table is visualized as depicted in Figure 9.

The tester now desires to edit a specific allergy record. From the actions column, the “Edit” button is selected as shown in Figure 10 and a form filled with data from this record shows up, as in Figure 11. The tester alters the type of allergy and clicks “Submit” again. This record has now been updated as shown in Figure 12.

Finally, the tester needs to delete a specific allergy record. In our test, we wanted to delete the last record. From the actions column, the “Delete” button is pressed and as we may notice in Figure 13 the record is removed from the data table.

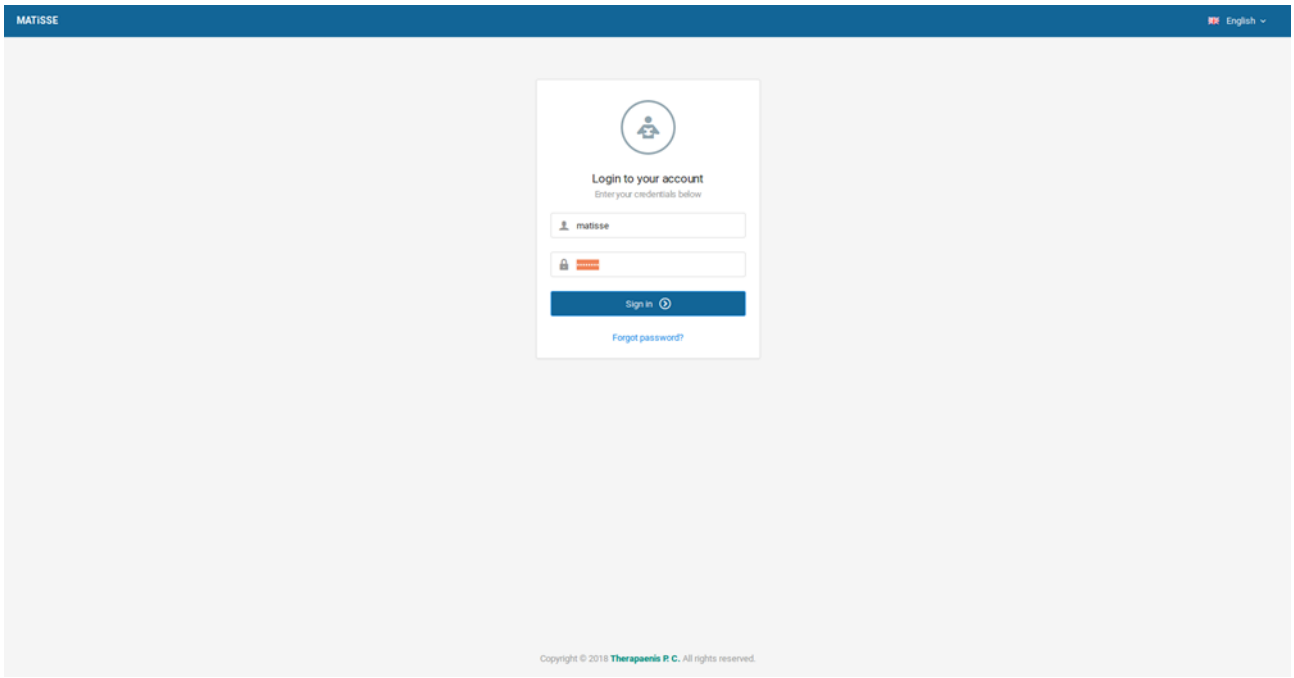


Figure 6: The user enters their credentials

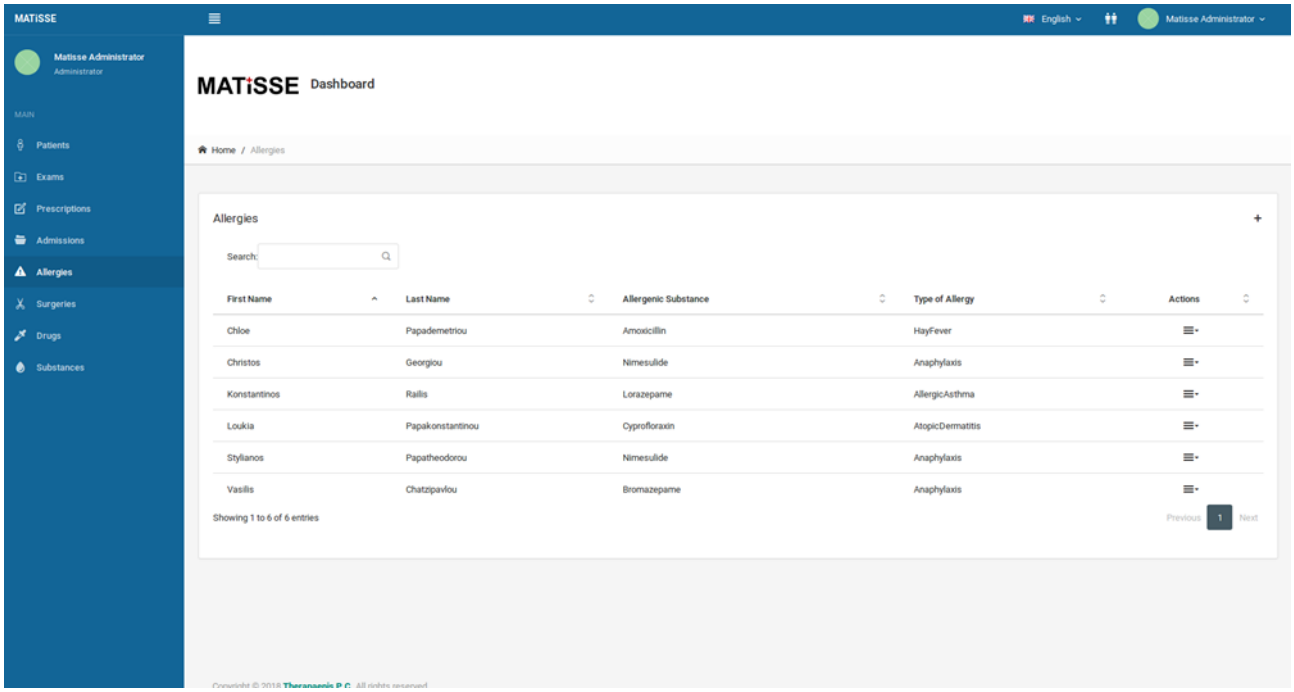


Figure 7: The signed in user visualizes a data table with Allergy records

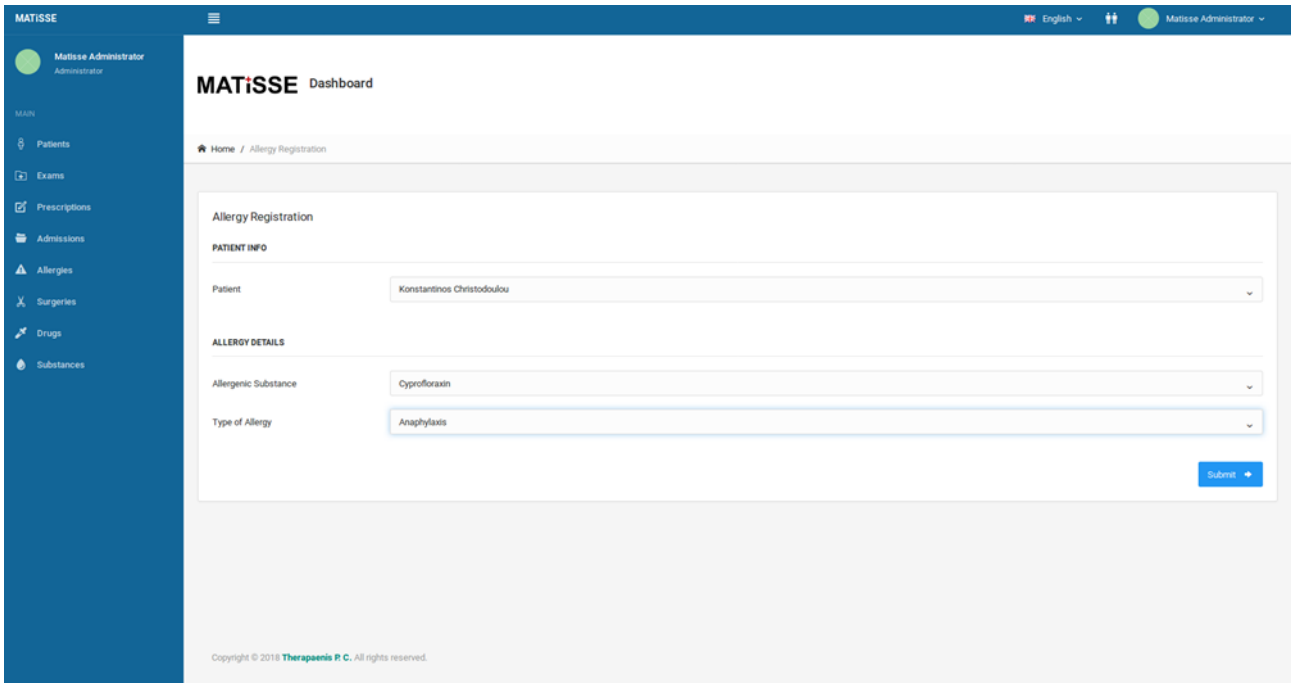


Figure 8: The patient fills the form to register a new allergy

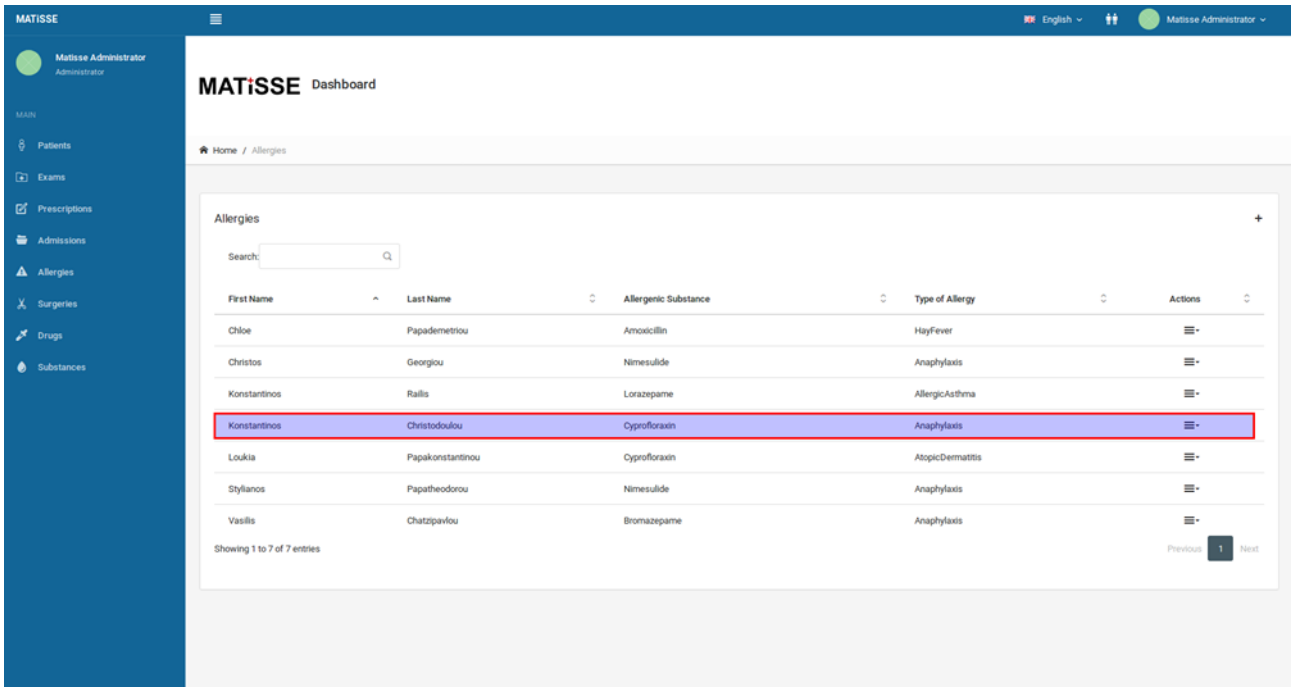


Figure 9: It is confirmed that the data table is refreshed after the addition of the new record

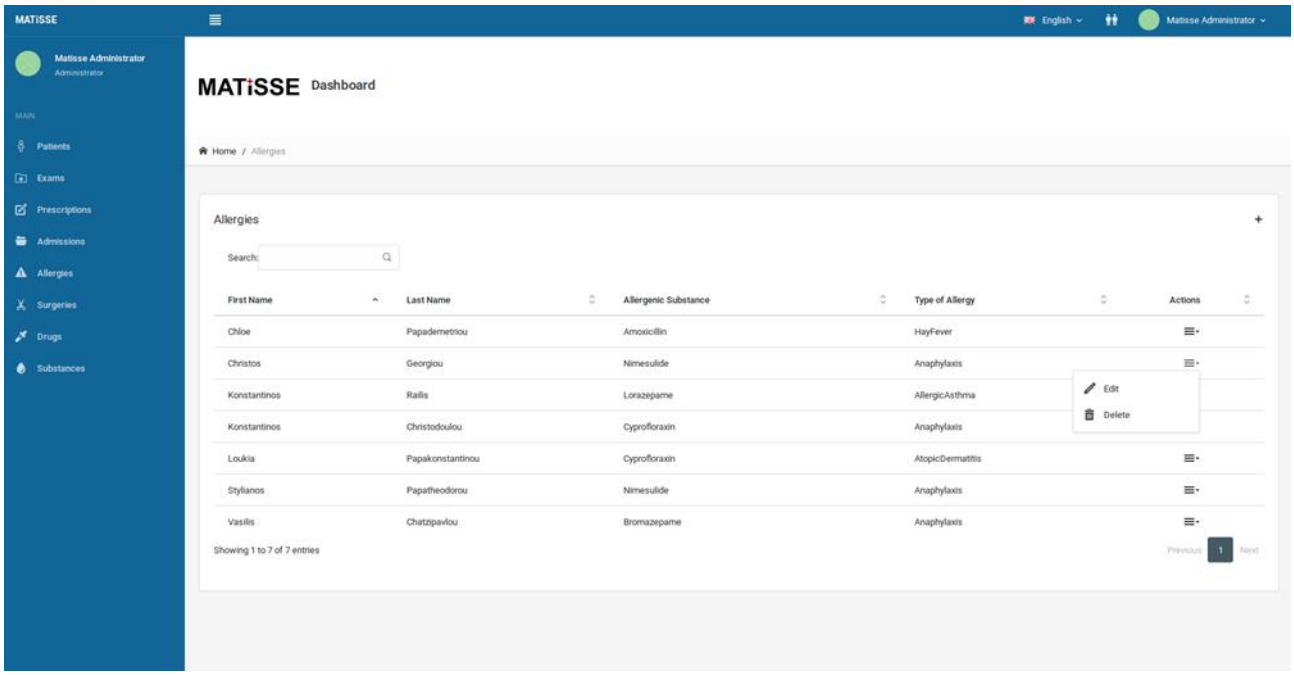


Figure 10: The available options that can be performed on a record

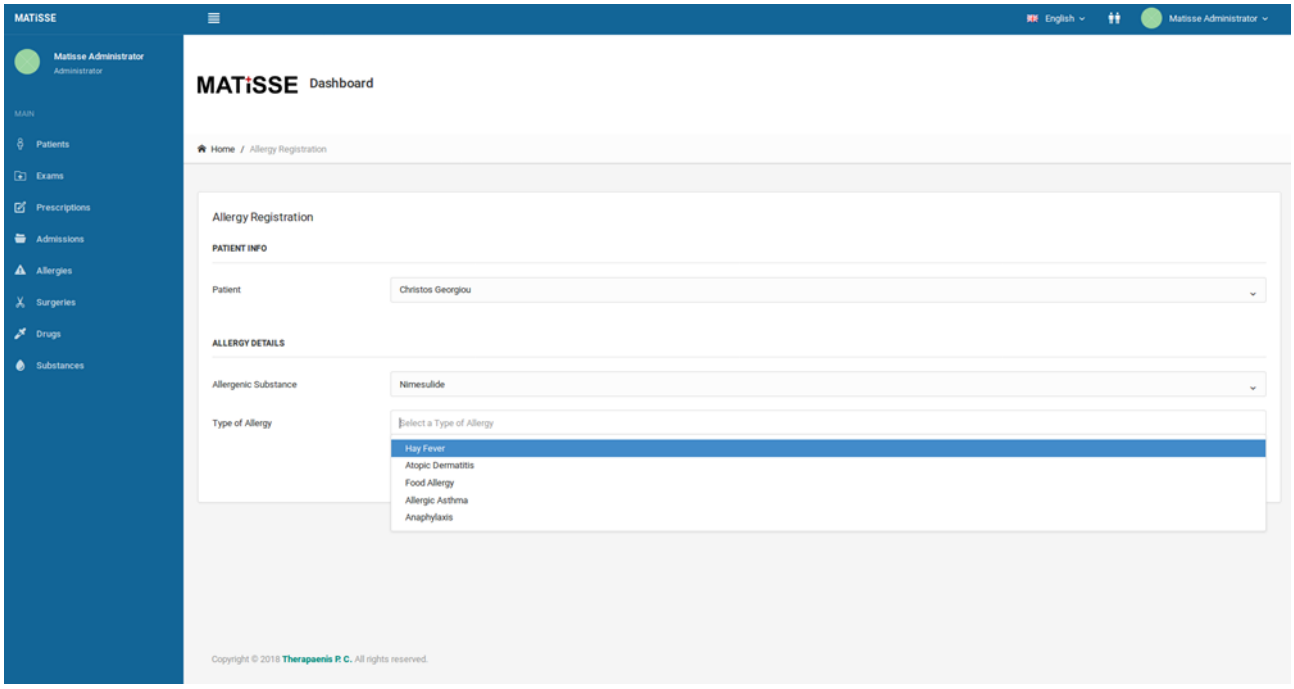


Figure 11: The user alters the "Type of Allergy" field

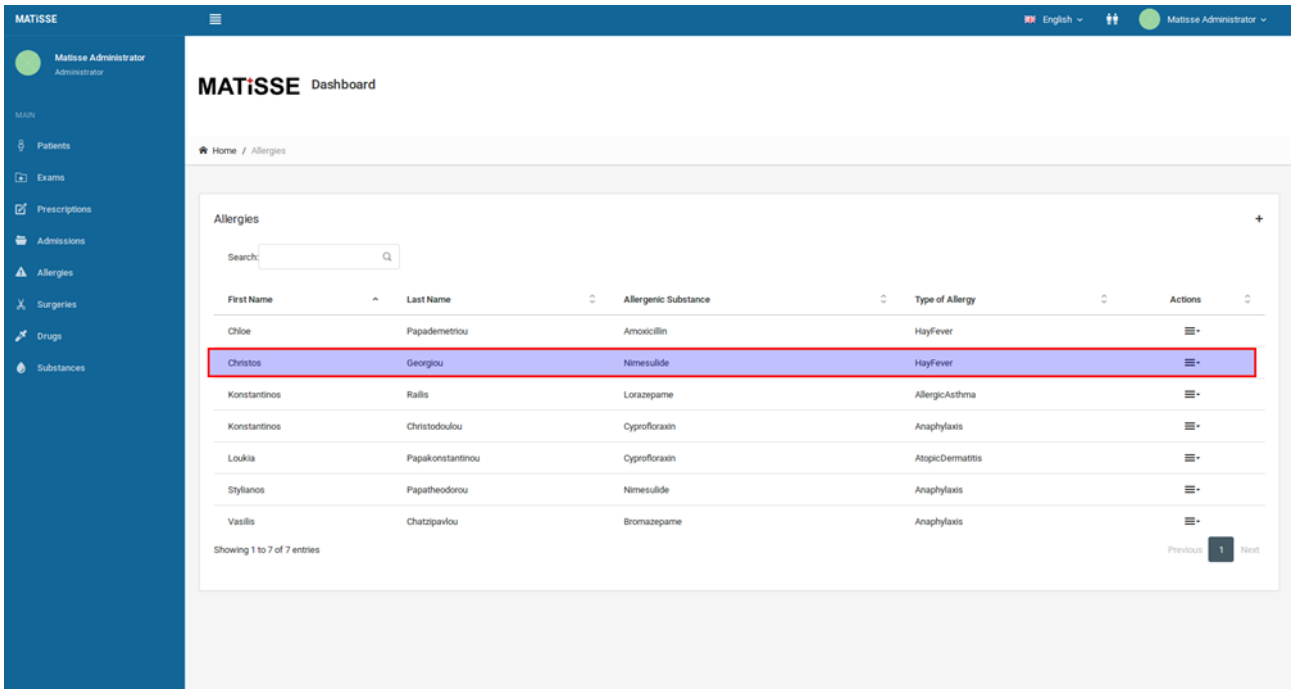


Figure 12: The record has been updated

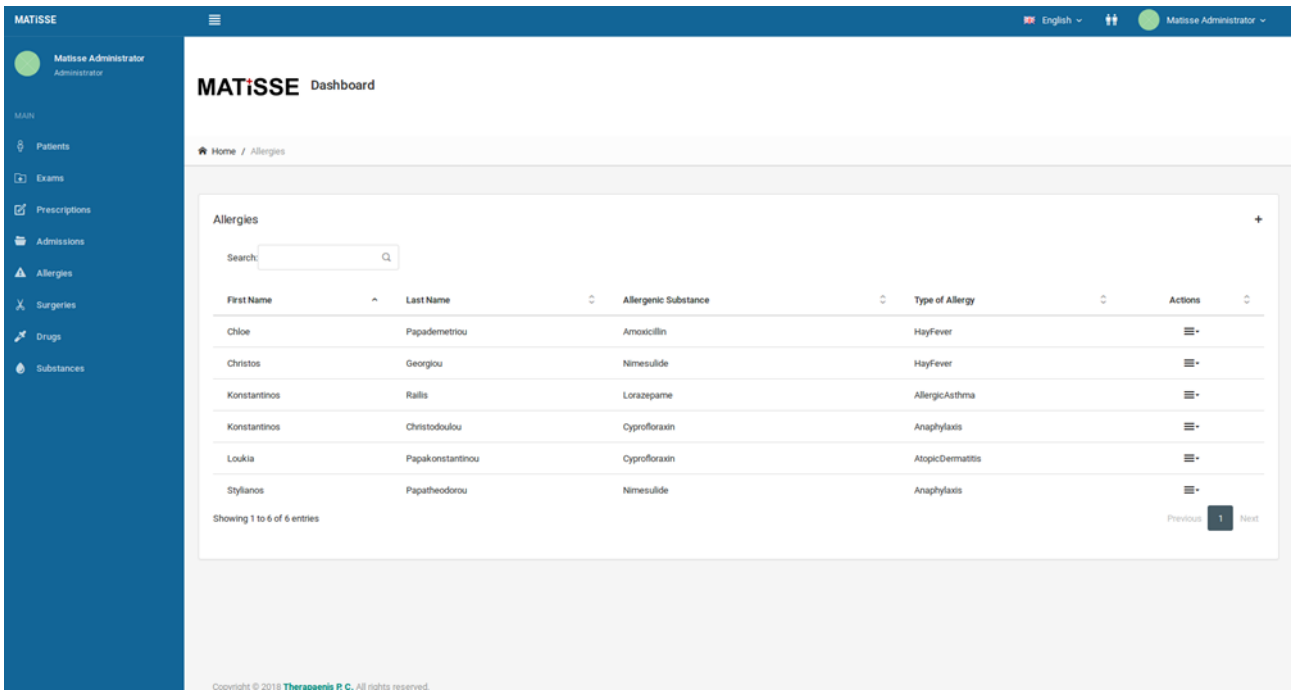


Figure 13: The last record has been deleted

4.2.9 Checkpoints

Checkpoint	Result	Comments
Is signing in necessary for accessing the dashboard?	PASS	All tests have been conducted with an Administrator account.
Is the user able to visualize the data available for all types of resources?	PASS	All types of resources (users, patients, exams, surgeries, allergies etc.) are successfully displayed.
Can the user fetch the form for a resource, in order to add a new record?	PASS	The user can add new data by clicking on the "+" symbol
Are the data tables updated automatically after the insertion of new data?	PASS	After clicking "Submit" on the form the visualized data are the updated ones.
Is the user able to edit a record of a resource type?	PASS	By clicking on "Edit" button from the "Actions" column of a data table.
Are the data tables updated automatically after the editing of a record?	PASS	After clicking "Submit" on the form the visualized data are the updated ones.
Is the user able to delete a record of a resource type?	PASS	By clicking on "Delete" button from the "Actions" column of a data table.
Are the data tables updated automatically after the deletion of a record?	PASS	After clicking "Delete" the visualized data are the updated ones.
Can the user sign out of the dashboard?	PASS	By clicking on its username on the top right corner and selecting "Logout".

4.3. MATISSE Smart Cart Testing

The MATISSE Smart Cart is an essential component for the Drugs' Stocktaking and In-Hospital Medication use cases, hence the testing of both its hardware and software utilities is of great significance. The Smart Cart comprises a Raspberry Pi3B+ which is responsible for the control of its visual indications as well as sounds. On the RPi, a REST API has been developed, and includes interfaces for testing the cart, shutting down the cart and receiving information about the indications that should be activated or deactivated.

4.3.1 Test Item

The item under test is the MATISSE Smart Cart component of MATISSE.

4.3.2 Objective

The test objective is to verify the correct operation of both software and hardware components that co-exist on the Smart Cart.

4.3.3 Features to be Tested

The following features are going to be tested:

- The operation of all LEDs
- The operation of all magnetic switches
- The operation of the LCD display
- The operation of the speakers
- The remote control of the above features through the API that RPi offers.

4.3.4 Test Prerequisites

- The power bank lying on the Smart Cart is charged.
- The RPi is configured to automatically connect to the network.

4.3.5 Test Description

1. The tester plugs the RPi on the power bank.
2. The tester notices a “Welcome to MATiSSE!” message on the LCD display after about 15 seconds that confirms that the RPi is ready to receive requests.
3. The tester sends a request to the /test_cart/ endpoint of the REST API and the LEDs of the Smart Cart blink once, one after the other.
4. The tester confirms that all LEDs of all drawers and bins have blinked.
5. The tester now sends a request to the /cart_control/ endpoint along with a JSON Object which is described in the next section, in order to switch a desired LED on and show the tester’s and hypothetical patient’s name on the LCD display.
6. The tester confirms that the tester’s and patient’s names appear on the LCD display, ensuring its expected operation.
7. The tester notices the outer LED of the drawer is on and confirms that it is the correct one.
8. The tester opens a different drawer.
9. The tester listens to the alarm that goes off, denoting that he/she has opened the wrong drawer.
10. The tester closes the wrong drawer and the alarm stops.
11. The tester confirms that the speakers and magnetic switches are operating correctly through the previous step.

12. The tester closes the wrong drawer and opens the correct one.
13. The tester notices that the LED he/she has requested is on.
14. The tester posts another JSON Object to the /cart_control/ endpoint in order to switch the LED off.
15. The tester repeats the process for all the Smart Cart's LEDs.
16. The tester shuts the Smart Cart down through the /shutdown/ endpoint that is offered by the REST API.
17. The tester confirms that the Smart Cart has shut down successfully.

4.3.6 Input

The input for the testing of the `/cart_control/` endpoint is a JSON Object. Assuming the tester wants to switch the 9th LED on in order to serve the corresponding patient which is assumed to be "Patient 1", the following JSON Object should be posted to the endpoint.

```
{
    "led_number": 9,
    "led_output": "True",
    "processor": "Tester",
    "patient": "Patient 1"
}
```

When the patient is served, the tester has to inform the Smart Cart to switch the LED off, so the following JSON Object is posted to the same endpoint.

```
{
    "led_number": 9,
    "led_output": "False"
}
```

4.3.7 Test Setup

The MATISSE components involved in this test are the following:

- Smart Cart Manager (SCM)
- Process Quality Manager (PQM)

4.3.8 Expected Result

After the RPi has initialized its operation, the tester notices a “Welcome to MATiSSE!” message on the LCD display. The expected result of the request to the /`test_cart/` endpoint is that all LEDs of all drawers and bins blink one by one, thus their correct operation from a hardware aspect is confirmed by the tester.

After a POST request sent to the `/cart_control/` endpoint with a JSON Object as shown in section 4.3.6, the tester observes that his/hers and the patient’s name appears on the LCD display, while the outer LED of the drawer in which the desired LED is switched on, is also on. On the opening of other drawers the alarm goes off and the tester validates the operation of the speakers and magnetic switches attached on the back of each drawer. On the opening of the correct drawer, the tester confirms that the requested LED is on.

The tester sends another POST request to the `/cart_control/` endpoint for switching the LED off and he confirms that the LED is switched off.

Last but not least, the tester confirms that the Smart Cart is shut down after sending a request to the /`shutdown/` endpoint.

4.3.9 Outcome

The RPi initializes its operation and a “Welcome to MATiSSE!” message appears on the LCD display (Figure 14). After a request to the /`test_cart/` endpoint all LEDs of all drawers and bins blink one by one, thus their correct operation from a hardware aspect is confirmed.

A POST request is sent to the /`cart_control/` endpoint with a JSON Object as shown in section 4.3.6. The tester observes that his/hers and the patient’s name appears on the LCD display, while the outer LED of the drawer in which the desired LED is switched on, is also on (Figure 15). On the opening of other drawers the alarm goes off and the operation of the speakers and magnetic switches attached on the back of each drawer is validated. On the opening of the correct drawer, the tester confirms that the requested LED is on (Figure 16).

The tester sends another POST request to the /`cart_control/` endpoint for switching the LED off and he confirms that the LED is switched off (Figure 17). Finally, the tester confirms that the Smart Cart is shut down after sending a request to the /`shutdown/` endpoint (Figure 18).

Figure 14: A welcome message appears on the screen

Figure 15: The tester's and patient's names appear on the LCD display and the requested bin's drawer LED is on

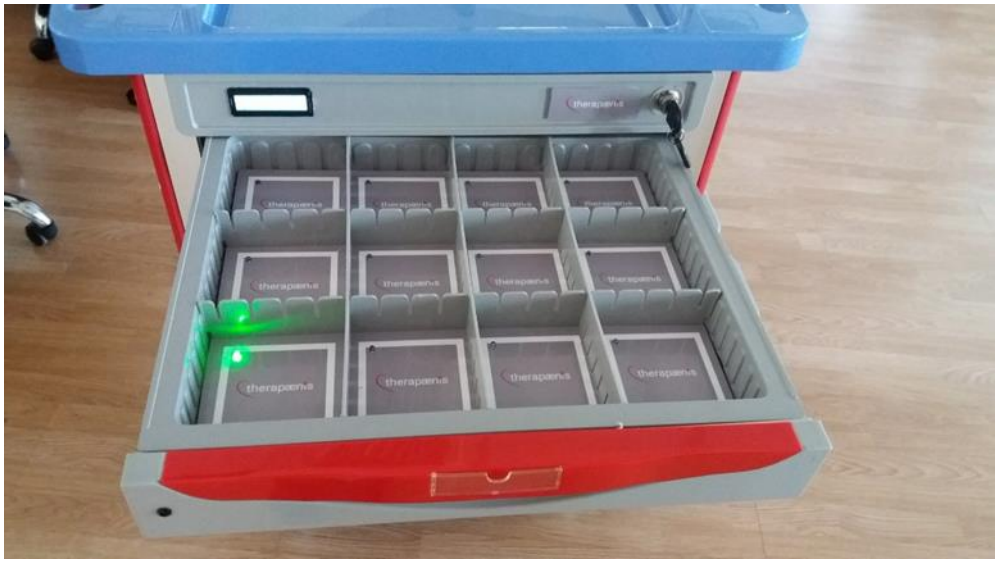


Figure 16: The requested bin's LED is on

Figure 17: The LED is switched off upon request

Figure 18: The Smart Cart is shut down after requested

4.3.10 Checkpoints

Checkpoint	Result	Comments
Does the Smart Cart initializes its operation after connection to the power bank?	PASS	Shortly after powering up the Smart Cart, a welcome message appears on the LCD display.
Do the LEDs of all drawers and bins work correctly?	PASS	The LEDs of all drawers and bins blink one by one as confirmed by the tester.
Can the tester request the Smart Cart to switch a LED on?	PASS	By issuing a POST request to the `/cart_control/` endpoint, including the LED number and other data.
Is the correct operation of the LCD display confirmed?	PASS	The tester's and patient's names sent in the previous step along with the LED number should be displayed.
Is the outer LED of the corresponding drawer switched on?	PASS	Depending on the led_number that has been sent previously, the outer LED of drawer (led_number div 12) + 1 should be on.

Does the alarm goes off when I open another drawer and not the one with its LED on?	PASS	The RPi is aware of an opening drawer via the magnetic switches on the back of the drawers.
Is the operation of the magnetic switches confirmed?	PASS	The operation is confirmed though the previous step.
Does the alarm stop when I close the wrong drawer?	PASS	The alarm should stop, ensuring that the RPi is aware of the drawer's closing through the magnetic switch.
Is the requested LED switched on when I open the correct drawer?	PASS	The RPi knows when I have opened the correct drawer, switches the outer LED off and the inner LED on.
Does the LED turn off upon request?	PASS	By sending a JSON Object to the `/cart_control/` endpoint.
Are all the LEDs working properly?	PASS	The previous steps were repeated for all drawers and bins.
Is the Smart Cart successfully shut down remotely?	PASS	By sending a request to `/shutdown/` endpoint.

4.4. MATISSE Mobile Application Testing

The MATISSE Mobile Application provides access to and allows the execution of the five MATISSE use cases by authorized hospital personnel through Android mobile devices, namely smart phones and tablets. The tests that are conducted aim at validating each of the following use cases and consequently the correct operation of the MATISSE Mobile Application component:

- Use Case 1: Drugs' Stocktaking
- Use Case 2: In-Hospital Medication
- Use Case 3: Smart Medical Exams
- Use Case 4: Pre-surgery Check
- Use Case 5: Patients' Monitoring

The first two use cases listed above will be performed sequentially as they interrelate, while the rest can be performed independently.

4.4.1 Drugs' Stocktaking

4.4.1.1 Test Item

The item under test is the "Drugs' Stocktaking" Use Case of MATISSE.

4.4.1.2 Objective

The test objective is to verify the correctness of the execution of a Drugs' Stocktaking session, during which the drugs for all patients are prepared and placed in the Smart Cart.

4.4.1.3 Features to be Tested

The following features are going to be tested:

- The login activity of the MATISSE Mobile Application.
- The operation of the Smart Tag Scanner (SSc) component.
- Scanning the cart's smart tags.
- The visualization of patients' and prescription doses' information on the mobile app screen.
- Scanning and identifying the drugs'.
- The combined operation of the MATISSE Mobile Application with the MATISSE Smart Cart via the MATISSE Back-End component.
- The integration of MATISSE with the EVRYTHNG platform.

4.4.1.4 Test Prerequisites

- At least one smart cart must be available
- An Android tablet or smart phone with the MATISSE mobile app installed must be available
- Both the smart cart and the mobile device must have adequate battery resources at the beginning of the process at the beginning of the process
- Communication between the smart cart, the MATISSE Mobile Application and the MATISSE Back-End must be available
- The hospital worker (e.g. nurse) must have a valid account with proper access rights in the Drugs' stocktaking use case
- Both the smart cart and the drugs must be attached with tags with QR codes. For drugs which must be stored cooled (between +2°C and +8°C), drugs' packaging must be attached with smart tags in the form of Time-Temperature Indicators (TTIs), sensitive in two threshold temperature values (upper and lower) to ensure that medication is stored between 2 and 8 degrees Celcius.
- A list of patients must be present in the MATISSE data storage
- A list of prescriptions for at least one patient must have been stored in the MATISSE Data Storage
- A list of drugs and drugs' packages must have been registered on the EVRYTHNG platform, at least the ones that are involved in prescriptions during the testing.

4.4.1.5 Test Description

During validating the "Drugs' stocktaking" use case, the tester will perform the following steps:

- The tester signs in the MATISSE mobile application.

- The tester selects the Drugs' stocktaking option
- The scanner is activated; the tester scans the cart
- The first patient appears with a list of medicine to be provided. At the same time, the led of the respective drawer is on. The tester opens the drawer. If it is wrong, a sound notifies him/her, so she/he closes it and opens the correct one. If it is correct, the led of the patient's bin is on.
- The tester selects a drug from the list in the mobile app, collects the drug (physically) and scans it. If it is wrong, the nurse is notified by the app. Also, the drug's storing conditions are verified through its smart tag in the mobile app. If storing conditions are fine, the drug is removed from the list, otherwise, the app waits for another item to be scanned. The process is repeated until the correct drug under proper storage conditions is collected.
- The tester assures that the value of "Number of Pills" property of this drug type and package has been reduced.
- The tester puts the drug in the patient's bin, the led of which is on.
- Steps 5-7 are repeated until the whole drug list is provided.
- The tester visualizes the next patient's medicine list in the mobile app and steps 4-8 are repeated for all patients.
- When all patients are served, the app returns to the welcome screen and the cart is ready.

4.4.1.6 Input

The input of this test is considered to be the Smart Cart's and the drugs' tags that are going to be scanned as described in the previous section.

4.4.1.7 Test Setup

The MATISSE components involved in this test are the following:

- Data Storage (DS)
- Access Manager (AM)
- Session Manager (SM)
- Smart Tag Scanner (SSc)

Additionally, in this test the EVRYTHNG platform is involved.

4.4.1.8 Expected Result

To begin with, the tester should be able to sign in the MATISSE Mobile Application with his/hers credentials. If a wrong or non-existent combination of username and password is given the tester should be notified with a related message. Having signed in the application, the tester should be able to select the "Drugs' Stocktaking" option and upon selection the QR scanner should be automatically initialized, prompting the tester to scan the Smart Cart's tag.

The tester follows the application's guidance and upon the scanning of the Smart Cart's tag, the application fetches the list of patients to prepare the drugs for. At this point all the patients with pending prescriptions are associated with a bin on the Smart Cart. The information and prescription doses list for the first patient appear on the screen. Simultaneously, the corresponding drawer's LED is on.

The tester should prepare the drugs for the first patient. He/She should select a drug from the list. The QR scanner launches automatically and urges the tester to scan the drug. If he scans a wrong drug, an alert window should be raised by the application notifying the tester for his/hers mistake. When he scans the correct one, the drug is removed from the list. At this point, the tester checks the EVRYTHNG platform to confirm that the value of "Number of Pills" property for this drug package has been reduced. The process is repeated for all the drugs in the list.

The tester should now place the prepared drugs in the corresponding patient's bin. The drawer's LED is already on. If the tester opens a wrong drawer an alarming sound should notify him/her. When he opens the correct one, the outer LED should switch off and the bin's LED should be on. The tester places the drugs in the bin, selects "NEXT PATIENT" and repeats the aforementioned process for all patients and their drugs in the list. Upon completion of the Drugs' Stocktaking session the mobile app should automatically return to the main menu.

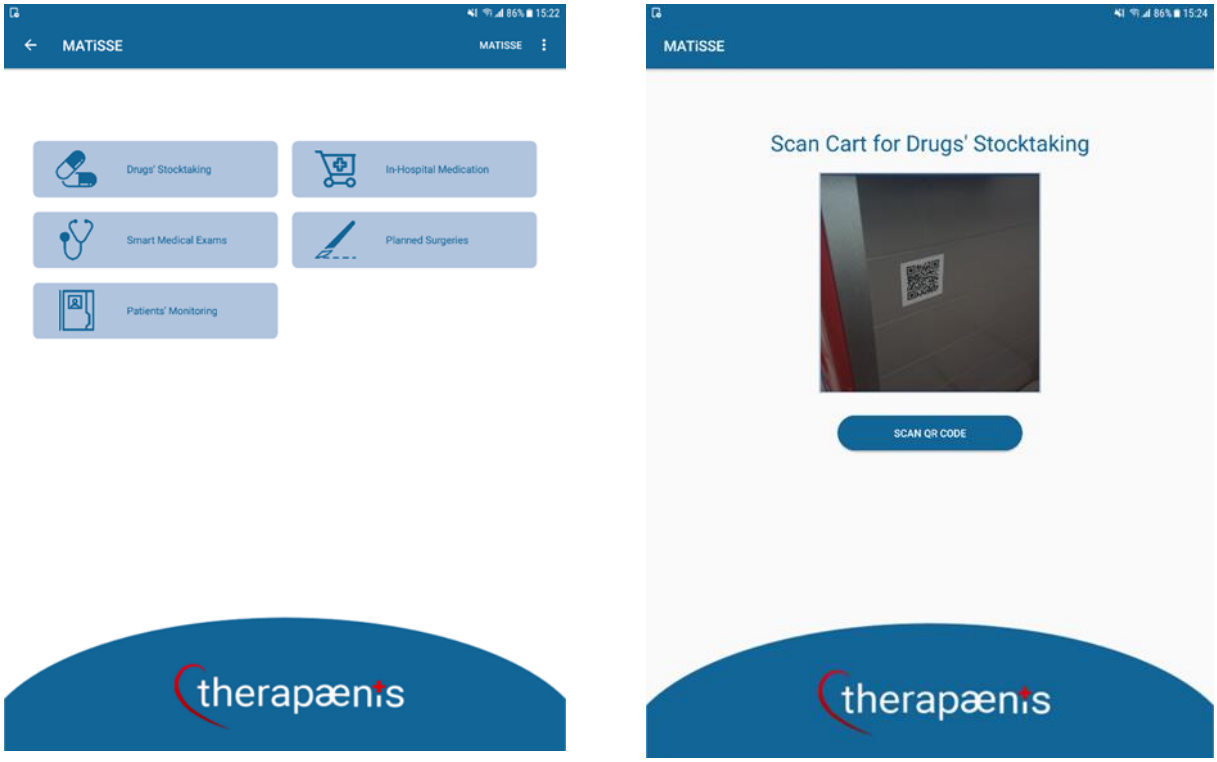
4.4.1.9 Outcome

To begin with, the tester launches the MATISSE Mobile Application and attempts signing in with a wrong set of credentials as shown in Figure 19. Then, he/she signs in with valid credentials and the main menu appears (Figure 20 (a)). Having signed in the application, the tester is able to select the "Drugs' Stocktaking" option and upon selection the QR scanner should be automatically initialized, and the tester scans the Smart Cart's tag as depicted in Figure 20 (b).

The application fetches the list of patients to prepare the drugs for. The information and prescription doses list for the first patient appear on the screen (Figure 21). Simultaneously, the corresponding drawer's LED is on (Figure 22). If the tester opens a wrong drawer an alarming sound really notifies him/her. When he opens the correct one, the bin's LED is on (Figure 23).

The tester prepares the drugs for the first patient. He selects Drug 2 from the list. The QR scanner launches automatically and urges the tester to scan the drug. He scans the correct one and the drug is removed from the list (Figure 25). Then he scans a wrong drug, an alert window is raised by the application notifying the tester for his mistake (Figure 26). He then selects and scans the correct one and the drug is removed from the list. The tester assures that the values of "Number of Pills" property for the corresponding drugs have been reduced. The tester places the prepared drugs in the corresponding patient's bin, then selects "NEXT PATIENT" and repeats the aforementioned process for all patients and their drugs in the list (Figure 28). Upon completion of the Drugs' Stocktaking session the mobile app returned to the main menu.

Figure 19: Signing in the MATISSE Mobile Application



(a) (b)

Figure 20: The main menu of the mobile application (a). The scanner is automatically activated and the Smart Cart's tag is scanned (b)

Figure 21: The applications has fetched the first patient with his prescriptions' doses

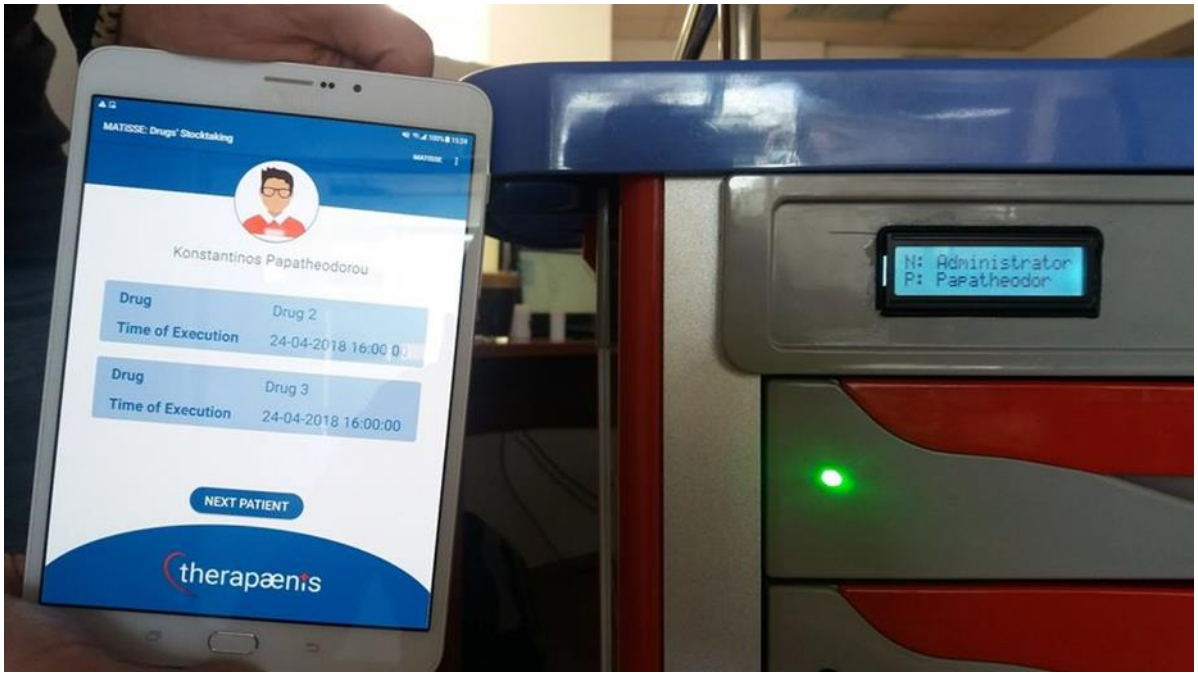


Figure 22: The patient is fetched and the corresponding drawer's LED in on. The patient's and tester's name appear on the LCD display

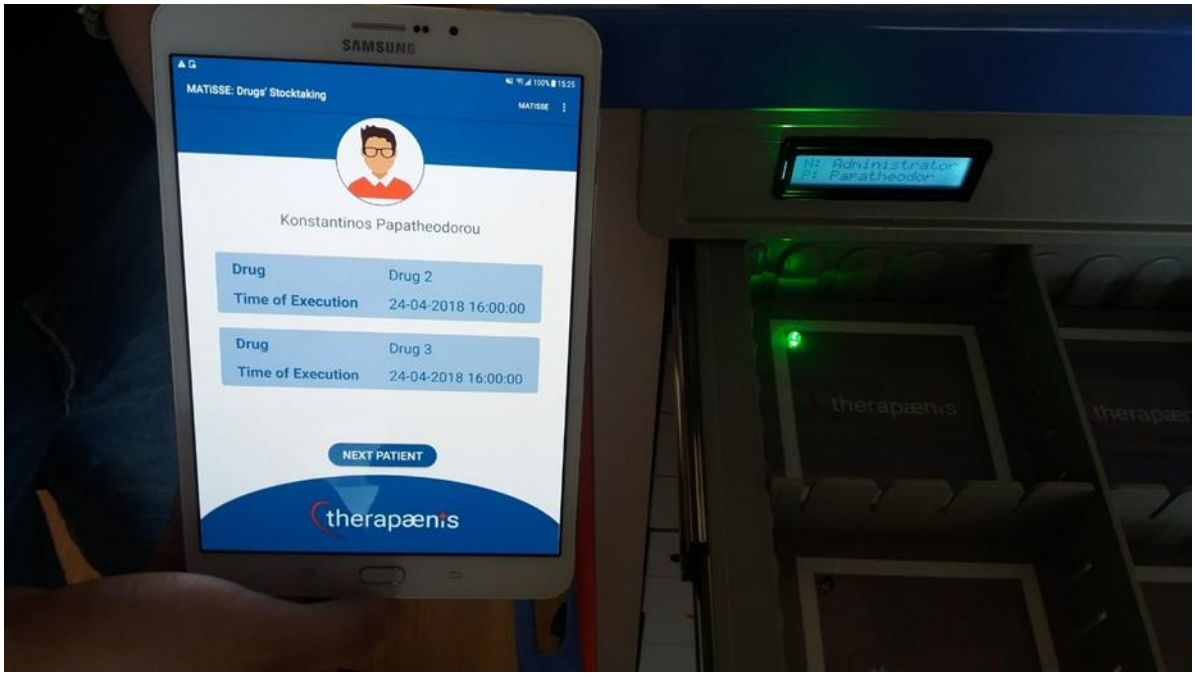


Figure 23: The tester opens the drawer and the corresponding patient's bin LED is on

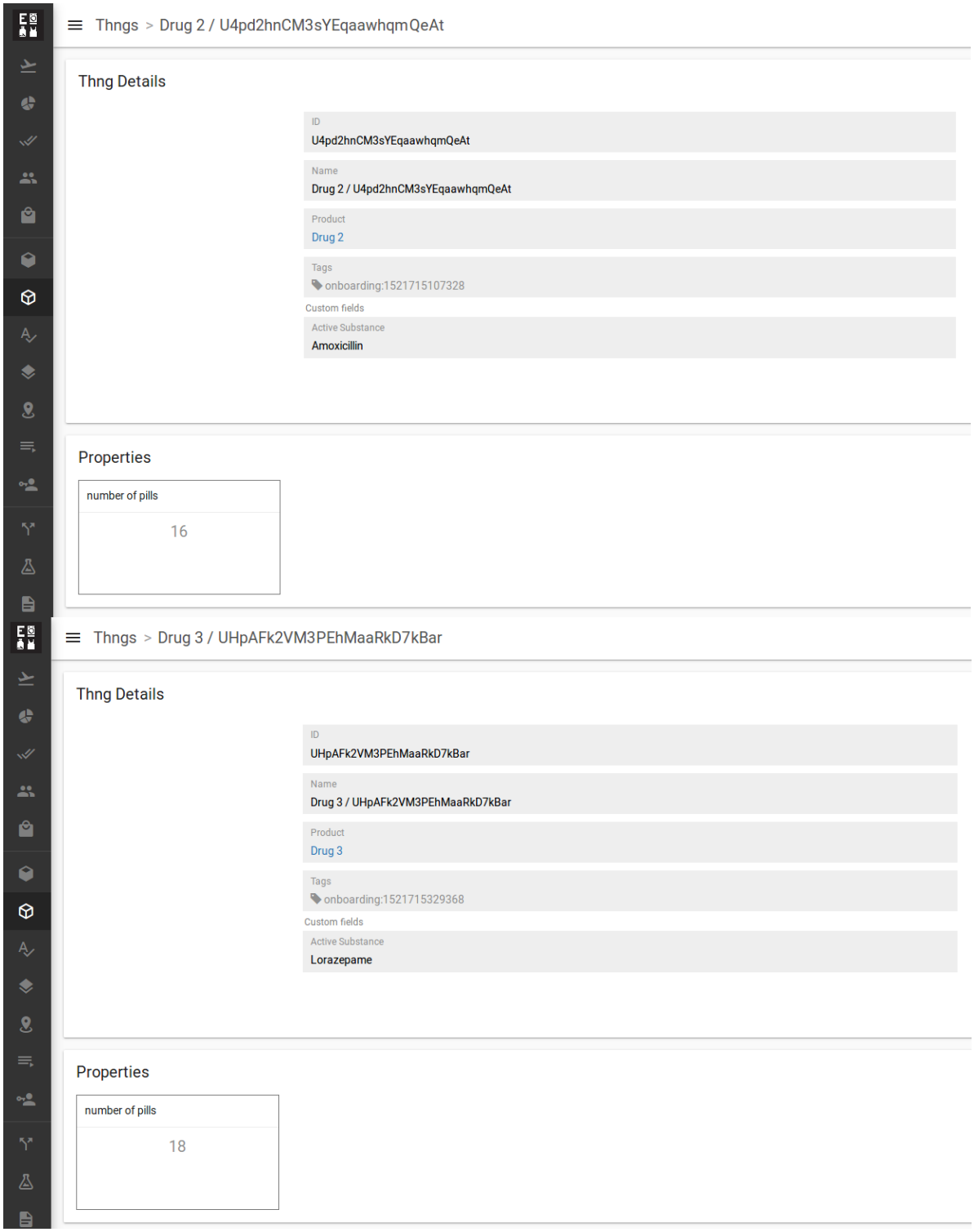


Figure 24: The drugs' packagings (thngs) with the initial values of "Number of Pills" properties

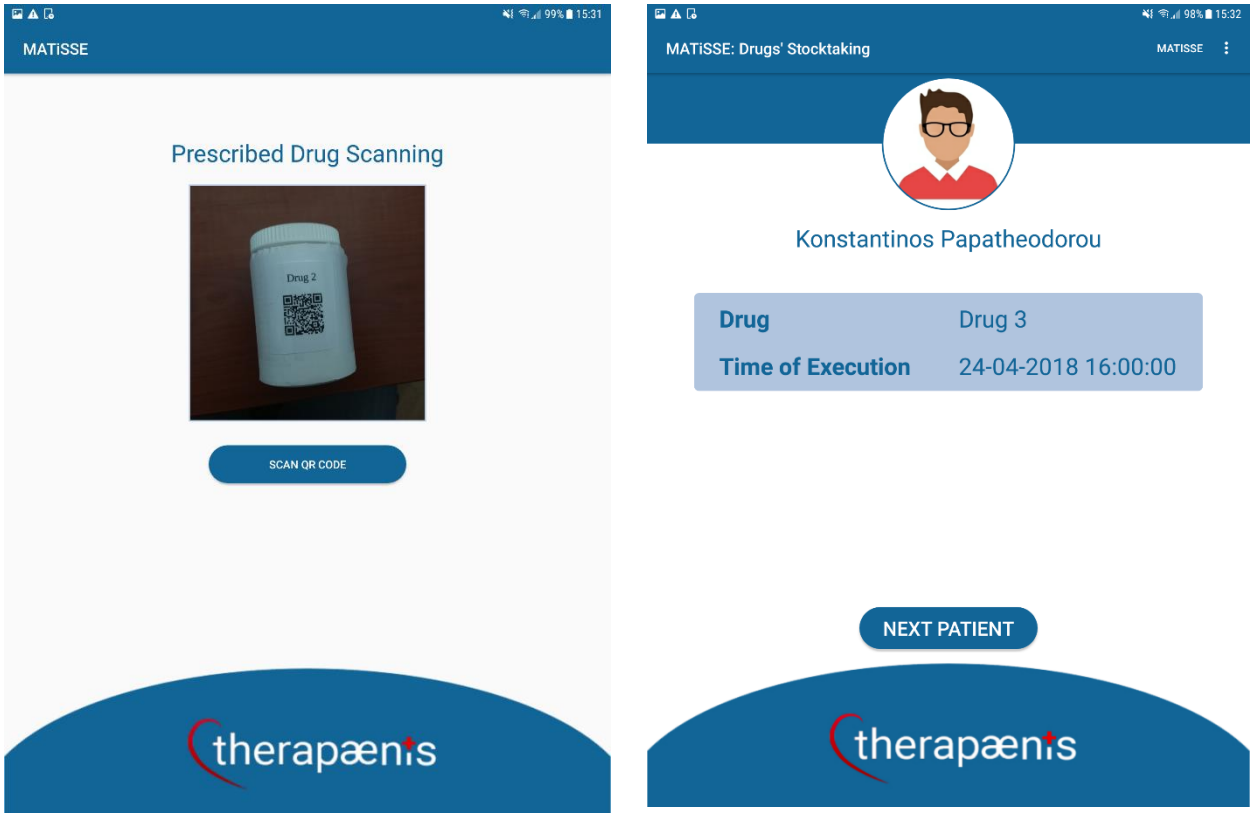


Figure 25: The tester selects Drug 2 and scans it. It is the correct so the dose is removed from the list.

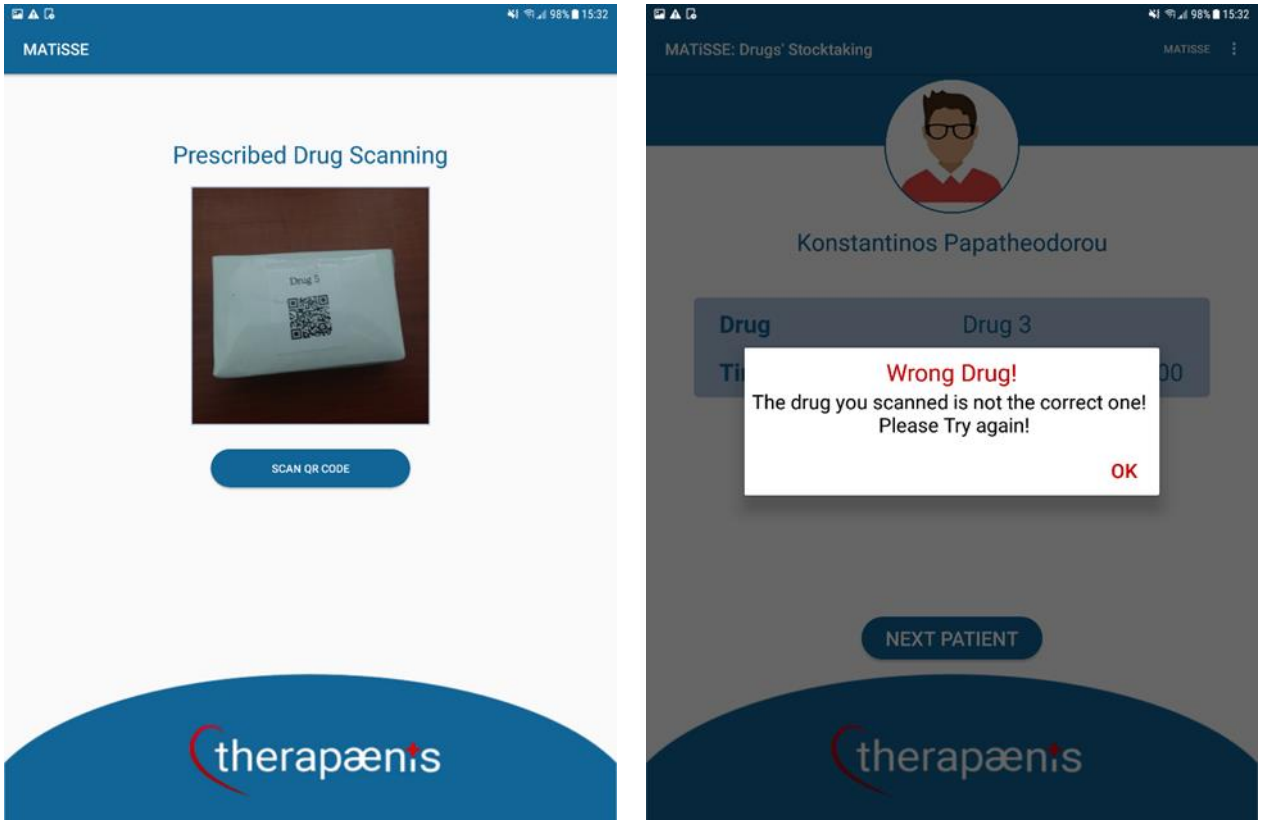


Figure 26: The tester should have scanned Drug 3 but, mistakenly, scans Drug 5. The application raises a related alert.

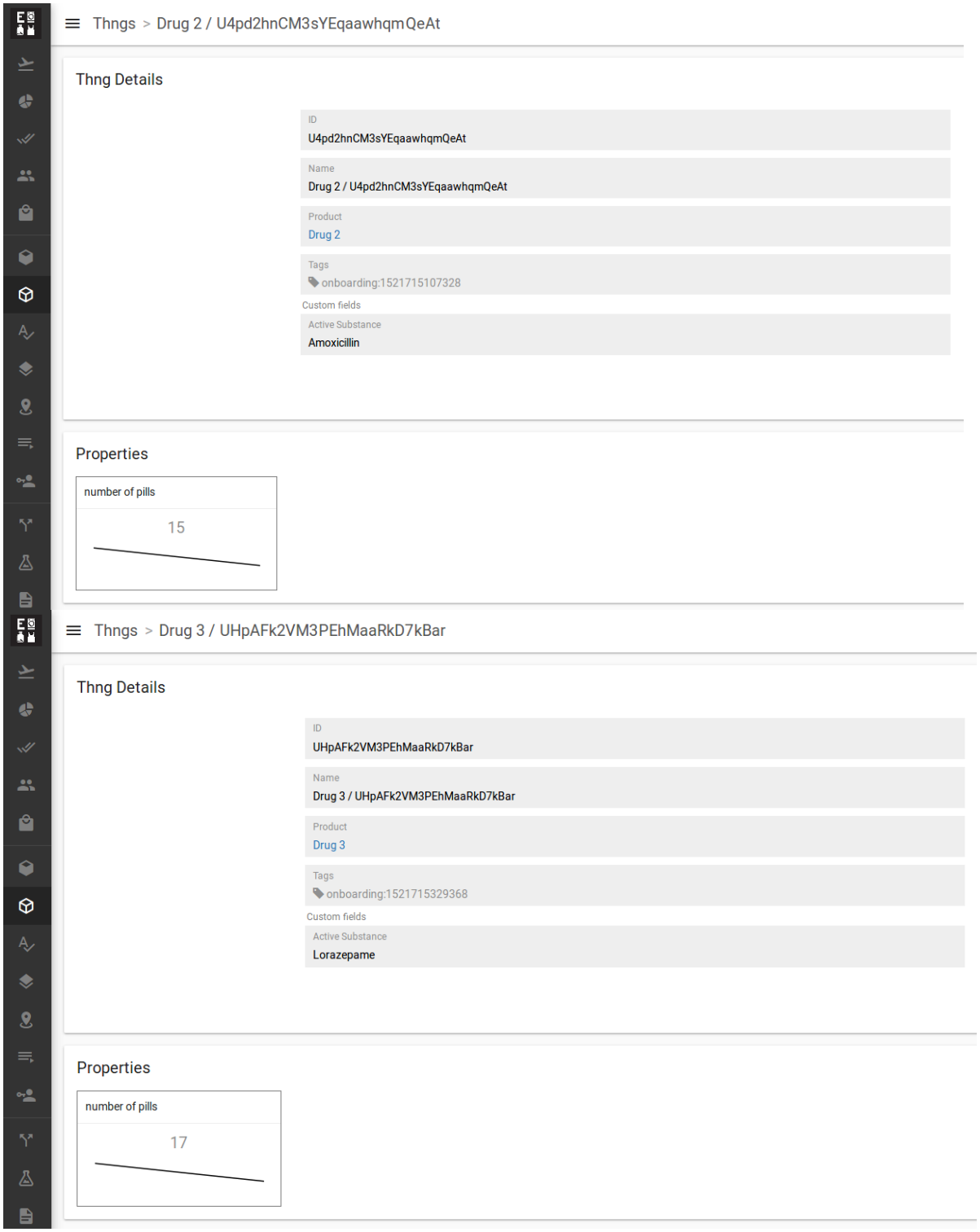


Figure 27: The values of "Number of Pills" property for the corresponding drugs are reduced

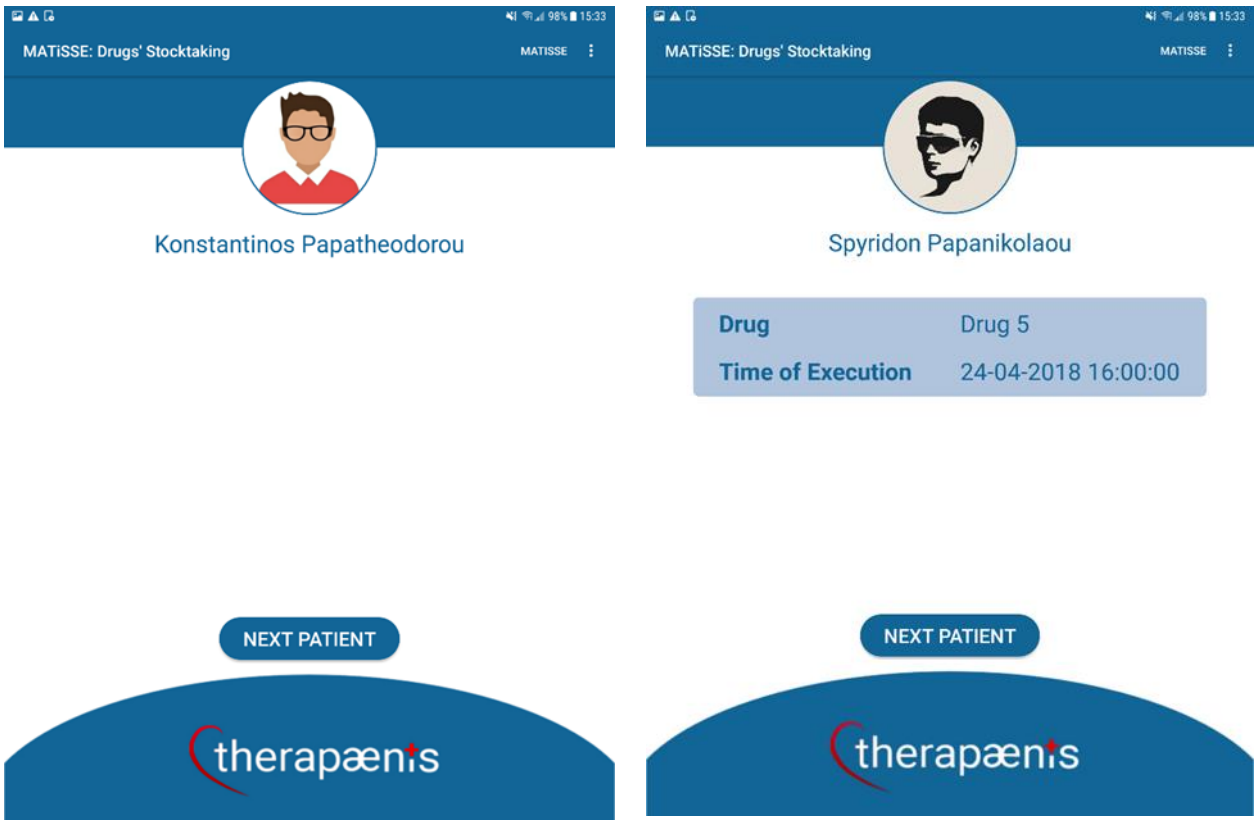


Figure 28: The current patient has no more prescribed drugs, the tester selects "NEXT PATIENT" and the next patient is fetched

4.4.1.10 Checkpoints

Checkpoint	Result	Comments
Is the tester able to sign-in with their credentials?	PASS	All tests have been conducted with an Administrator account.
In case the inserted username and/or password are wrong, is the user notified accordingly?	PASS	An invalid username or password message appears when attempting to sign in with wrong credentials.
Is it possible to navigate in any app activity without signing in?	PASS	Navigating in the application is impossible without signing in.
Is the user automatically signed out after some time of inactivity?	PASS	If the access token is expired or invalidated the user is obliged to sign in again.
Is it possible to select the "Drugs' stocktaking" option?	PASS	After having already signed in the mobile application.
Is the QR scanner automatically activated?	PASS	On the selection of "Drugs' Stocktaking" option, the scanner is activated and urges the tester to scan the Smart Cart's tag.
Does the cart screen display the user and current patient?	PASS	When the application fetches the requested information, the cart's LCD display already

		shows the tester's and current patient's names.
Does any patient name appear after the cart scan?	PASS	The patient's name and photo appear on the mobile app screen.
Does the list of drugs for this patient appear?	PASS	Along with the patient's information.
Is the cart drawer LED on?	PASS	The outer LED of the corresponding drawer is on.
In case the wrong drawer is opened, does a sound notify the tester?	PASS	A discrete alarming sound notifies the tester.
In case the right drawer is opened is its LED off?	PASS	If the correct drawer is opened the outer LED is off.
In case the right drawer is opened is its bin's LED on?	PASS	When the outer LED switches off, the bin's LED is on.
From the drugs' list, can every drug be selected?	PASS	The tester is able to select all drugs.
Does the scanner open automatically?	PASS	On clicking of a drug, the scanner opens automatically.
Is it easy to understand that the drug has to be scanned?	PASS	A related message is displayed that urges for the drug's scanning.
In case a wrong drug is scanned, is it easy for the user to understand it?	PASS	An alert window is raised that notifies about the scanning of a wrong drug.
When the correct drug is scanned, does it disappear from the list?	PASS	The drug is removed from the list.
When all drugs are scanned, is it easy for the user to understand that the next patient option should be selected?	PASS	The "NEXT PATIENT" button makes it clear that should be pressed when the drugs of a patient are successfully prepared.

4.4.2 In-Hospital Medication

4.4.2.1 Test Item

The item under test is the "In-Hospital Medication" Use Case.

4.4.2.2 Objective

The test objective is the verification of an In-Hospital Medication session, during which all the drugs existent in the Smart Cart, as a result of Drugs' Stocktaking, are delivered to the corresponding patients.

4.4.2.3 Features to be Tested

The following features are going to be tested:

- The login activity of the MATISSE Mobile Application.
- The operation of the Smart Tag Scanner (SSc) component.
- Scanning the patients' smart tags.

- Visualization of patients' information on the mobile app screen.
- The combined operation of the MATISSE Mobile Application with the MATISSE Smart Cart via the MATISSE Back-End component.

4.4.2.4 Test Prerequisites

- At least one smart cart must be available, already equipped with patients' drugs, following use case 1 process.
- An Android tablet or smart phone with the MATISSE mobile app installed must be available.
- Both the smart cart and the mobile device must have adequate battery resources at the beginning of the process.
- Communication between the smart cart, the MATISSE mobile app and the MATISSE Back-End must be available.
- The hospital worker (e.g. nurse) must have a valid account with proper access rights in the In-Hospital Medication use case.
- Both the smart cart, the drugs and the patients must carry tags with QR codes.

4.4.2.5 Test Description

During the validation of the "In-Hospital Medication" Use Case, the tester will accomplish the following steps:

1. The tester signs in the MATISSE mobile application.
2. The tester selects the "In-Hospital Medication" option.
3. The mobile app's QR scanner is activated and the tester scans the first patient's tag.
4. The patient's data appear in the mobile app screen, including the patient's photo and an overview of the cart's drawers, indicating the drawer where the current patient's drugs are stored. At the same time, the drawer's led is on.
5. The tester selects the drawer through the app and the position of the current patient's bin appears.
6. The tester opens the drawer of the Smart Cart, the led of which is on. If she/he opens the wrong drawer, a sound will notify her/him. If she/he opens the correct one, the current patient's bin led is on.
7. The tester takes the drugs and selects the bin from the app.
8. The tester confirms that the bin's led is off.
9. The tester repeats tests 2-8 until all the patients are provided with their medication.

4.4.2.6 Input

The input of this test is considered to be the patient's tags that are going to be scanned as described in the previous section.

4.4.2.7 Test Setup

The MATISSE components involved in this test are the following:

- Data Storage (DS)
- Access Manager (AM)
- Session Manager (SM)
- Smart Tag Scanner (SSc)

4.4.2.8 Expected Result

To begin with, the tester signs in MATISSE Mobile Application and selects the In-Hospital Medication option. The QR scanner should launch automatically and urge the tester to scan the patient he/she desires to deliver his prescribed drugs to.

The tester scans a patient's tag. The application should then fetch the patient's name and photo along with a front view of the Smart Cart, denoting the drawer in which this patient's drugs are stored. At the same time, the LED of this drawer on the Smart Cart should be on and the tester should be able to confirm that it is the same drawer. The tester selects the drawer and the application shows an overview of this drawer's bins, denoting the correct one again.

Then, the tester should open the drawer on the cart. If he opens a wrong one, a sound will notify him/her. When the correct one is opened, the outer LED should switch off and the bin's LED should be on. The tester confirms it is the same bin as the one on the mobile app. The drugs are delivered to the patient. Finally, the tester should select the bin through the mobile app's screen and the bin's LED will be switched off.

The aforementioned process is repeated until all drugs that are existent on the Smart Cart are delivered to the corresponding patients.

4.4.2.9 Outcome

To begin with, the tester signed in MATISSE Mobile Application and selected the In-Hospital Medication option. The QR scanner launched automatically and the tester scanned the patient's tag for drug delivery (Figure 29).

The application fetched the patient's name and photo along with a front view of the Smart Cart, denoting the drawer in which this patient's drugs are stored. At the same time, the LED of this drawer on the Smart Cart is on (Figure 30). The tester confirmed that the drawers are the same. The tester selected the drawer and the application shows an overview of this drawer's bins, denoting the correct one again. Then the tester opens a wrong drawer and a sound notified him. He opens the correct drawer and the bin's LED was on. The tester confirmed it was the same bin as the one on the mobile app (Figure 31). The hypothetical drugs are delivered to the patient. The tester selected the bin through the mobile app's screen and the bin's LED is switched off (Figure 32).

The aforementioned process is repeated until all drugs that are existent on the Smart Cart are delivered to the corresponding patients.

Figure 29: The QR scanner launched automatically when the In-Hospital Medication option was selected.

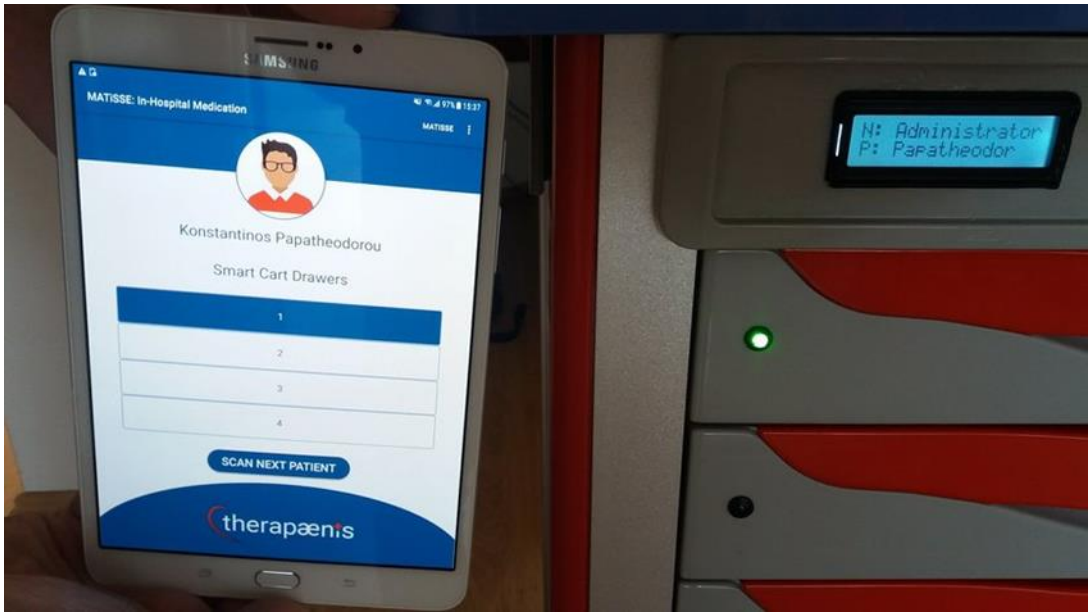


Figure 30: The application fetches the scanned patient and an overview of the cart. It is confirmed the drawer from the app and the drawer with the LED on are the same.

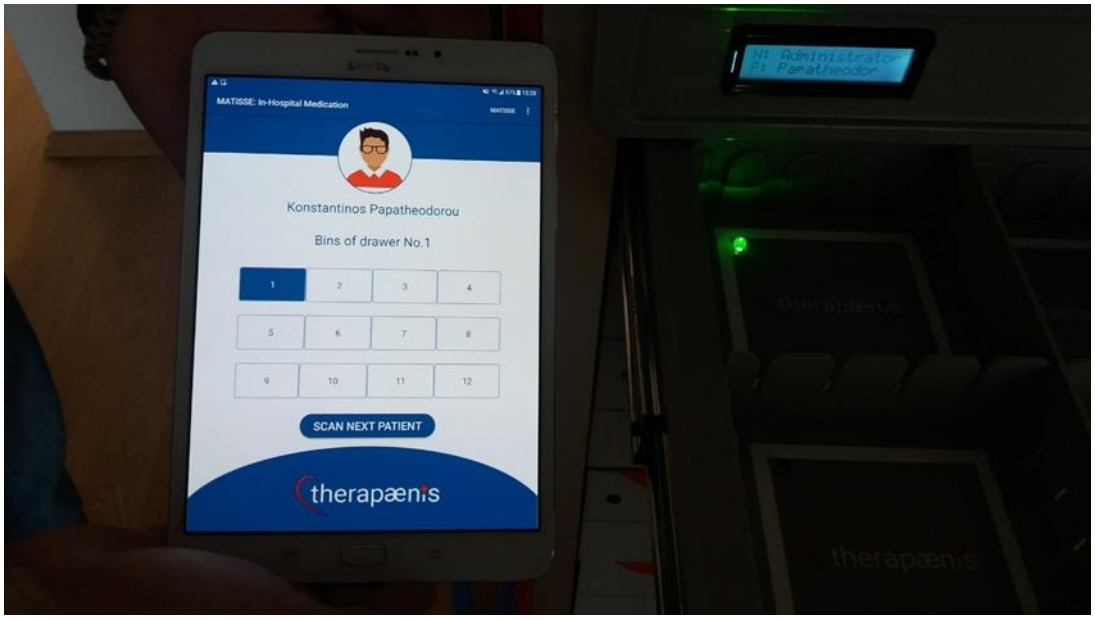


Figure 31: The tester selected the drawer from the app's screen, opened the correct drawer on the smart cart and confirms that the bin denoted from the app is the same as the one with the LED on, on the cart

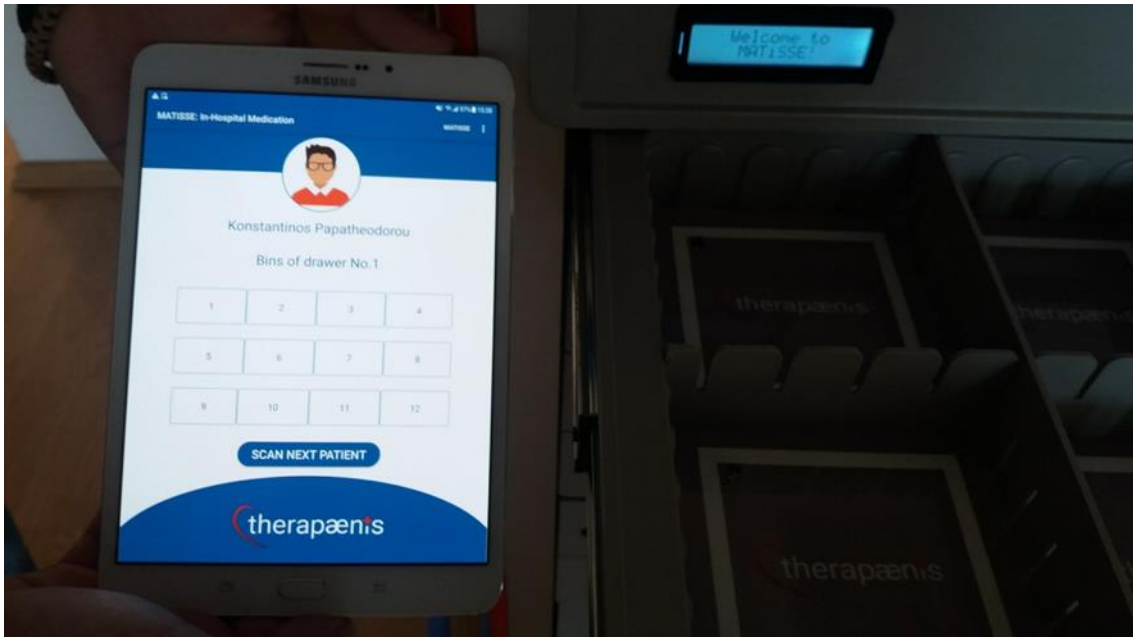


Figure 32: The tester selects the bin through the app, and the corresponding LED on the Smart Cart switches off

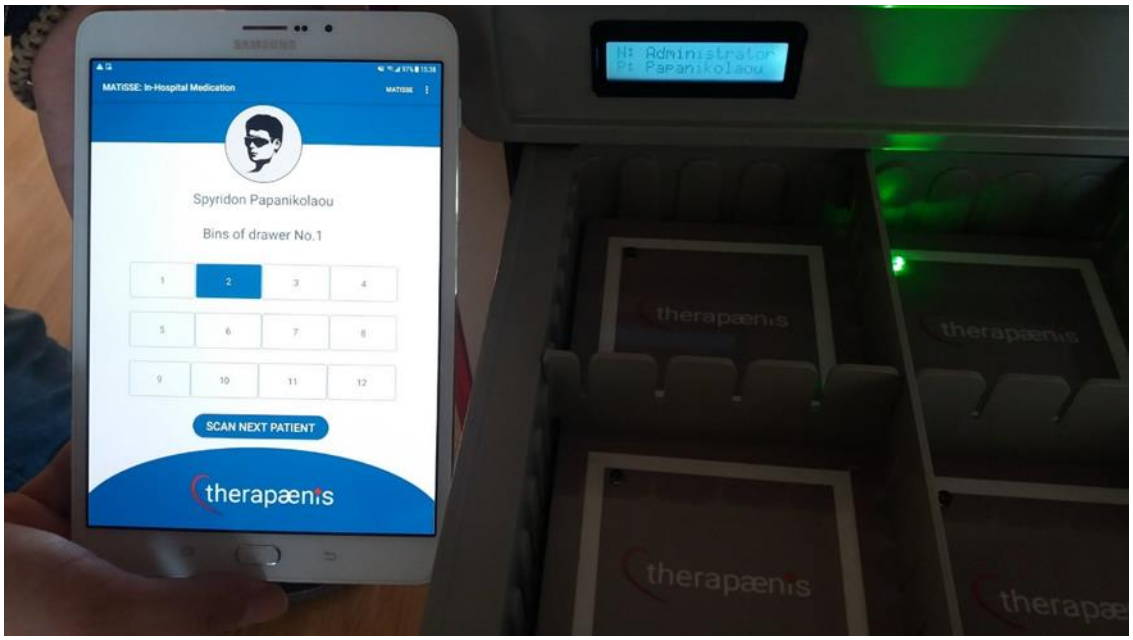


Figure 33: The tester selects "SCAN NEXT PATIENT", scans another patient and the application fetches the data as in the previous patient.

4.4.2.10 Checkpoints

Checkpoint	Result	Comments
Is it possible to select the “In-Hospital Medication” option?	PASS	All tests are conducted using an Administrator account.
Is the QR scanner automatically activated?	PASS	When the “In-Hospital Medication” option is selected the scanner is automatically activated.
Is it easy for the tester to understand that the patient should be scanned?	PASS	A message is displayed that urges the tester to scan the patient for the In-Hospital Medication session.
Is the QR scanning possible?	PASS	When clicking on the “Scan QR Code” button.
Is it easy for the tester to understand and verify that the patient’s drugs are in the correct bin?	PASS	The application fetches the patient’s name and photo as well for cross-reference.
In the smart cart, is the correct drawer led on (if not already opened)?	PASS	The corresponding drawer’s LED is on.
In case the wrong drawer is opened, is there a sound notification for the tester?	PASS	A discrete alarming sound notifies the tester.
Is the correct bin’s led on?	PASS	If/When the correct drawer is opened, the bin’s LED is off.
Is it possible for the tester to select/verify the drawer and the bin through the mobile app?	PASS	The application fetches an overview of the Smart Cart, denoting the correct drawer and bin that the current patient’s drugs are stored.
Does the next patient appear?	PASS	A patient appears only when he/she is scanned.

4.4.3 Smart Medical Exams

4.4.3.1 Test Item

The item under test is the “Smart Medical Exams” Use Case.

4.4.3.2 Objective

The test objective is the verification of a Smart Medical Exams session, in which a clinician can scan a specific patient and fetch information about his exam record.

4.4.3.3 Features to be Tested

The following features are going to be tested:

- The login activity of the MATISSE Mobile Application.

- The operation of the Smart Tag Scanner (SSc) component.
- Scanning the patients' smart tags.
- Visualization of patients' and their exams' information on the mobile app screen.

4.4.3.4 Test Prerequisites

- An Android tablet or smart phone with the MATISSE mobile app installed must be available;
- The mobile device must have adequate battery resources at the beginning of the process;
- Communication between the MATISSE mobile app and the MATISSE backend must be available;
- The hospital worker (e. g. doctor) must have a valid account with proper access rights in the Smart Medical Exams use case;
- At least one patient and at least one associated exam record must be stored in the MATISSE Data Storage;
- The patient(s) involved must carry tag(s) with QR codes.

4.4.3.5 Test Description

During the validation of this use case, the tester must follow the steps below:

1. The tester signs in the MATISSE Mobile Application.
2. The tester selects the Smart Medical Exams option.
3. The mobile app's QR scanner is activated and the tester scans the patient's tag.
4. The tester visualizes the list of available exams for the scanned patient, if any, through the mobile app.
5. The tester selects one medical exam item and visualizes details about it.

4.4.3.6 Input

The input of this test is considered to be the patient's tags that are going to be scanned as described in the previous section.

4.4.3.7 Test Setup

The MATISSE components involved in this test are the following:

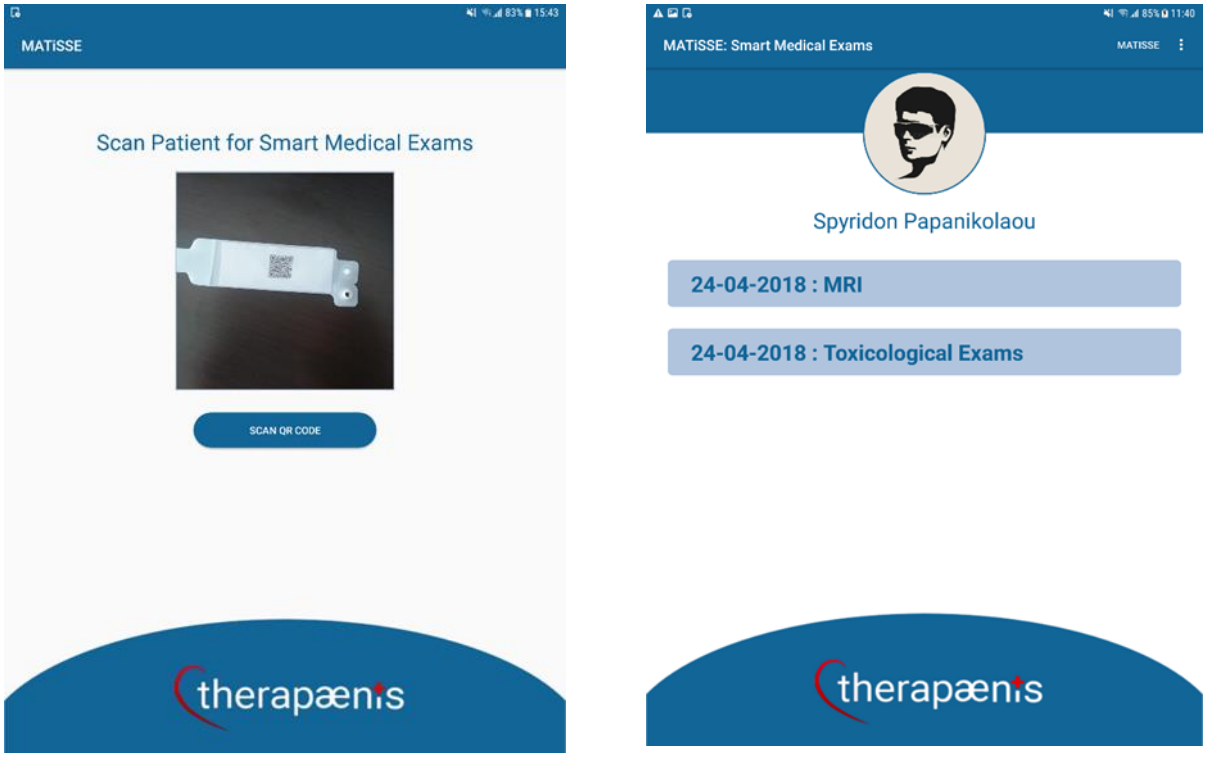
- Data Storage (DS)
- Access Manager (AM)
- Session Manager (SM)
- Smart Tag Scanner (SSc)

4.4.3.8 Expected Result

Firstly, the tester signs in the MATISSE Mobile Application and selects the Smart Medical Exams option. The QR scanner automatically launches and urges the tester to scan the patient's tag for a Smart Medical Exams session. The tester scans the patient's tag and, if existent, his/hers exam record appears on the screen along with his/hers name and photo.

4.4.3.9 Outcome

The tester signed in the MATISSE Mobile Application successfully and selected the Smart Medical Exams option. The QR scanner launched automatically and the tester scanned a patient's tag for a Smart Medical Exams session (Figure 34 (a)). The patient's exam record appears on the screen along with his/her name and photo. (Figure 34 (b)). The tester can select an exam and visualize its details (Figure 35).



(a) *(b)*

Figure 34: The patient's tag is scanned (a) and his exam record is visualized (b) on the screen

Figure 35: The tester visualizes specific exams

4.4.3.10 Checkpoints

Checkpoint	Result	Comments
Is it possible to select the “Smart Medical Exams” option?	PASS	The tester has already signed in with an Administrator account.
Is the QR scanner automatically activated?	PASS	After the selection of “Smart Medical Exams” option.
Is the QR scanning possible?	PASS	After clicking the “Scan QR Code” button the scanning is enabled.
In case there are exams for the patient, is information provided?	PASS	The patient data along with the recorded exams are fetched by the application.
In case there are no exams for the patient, is the user notified accordingly?	PASS	The patient’s data are fetched but no exam is displayed.

4.4.4 Pre-surgery Check

4.4.4.1 Test Item

The item under test is the “Pre-surgery Check” Use Case.

4.4.4.2 Objective

The test objectives is the verification of a Pre-surgery Check session, in which a clinician can scan a specific patient and fetch information about a scheduled surgery.

4.4.4.3 Features to be Tested

The following features are going to be tested:

- The login activity of the MATISSE Mobile Application.
- The operation of the Smart Tag Scanner (SSc) component.
- Scanning the patients' smart tags.
- Visualization of patients' and their surgeries' information on the mobile app screen.

4.4.4.4 Test Prerequisites

- An Android tablet or smart phone with the MATISSE mobile app installed must be available.
- The mobile device must have adequate battery resources at the beginning of the process.
- Communication between the MATISSE mobile app and the MATISSE backend must be available.
- The hospital worker (e.g. doctor or nurse) must have a valid account with proper access rights in the Planned Surgeries use case
- At least one patient and at least one associated surgery record must be stored in the MATISSE Data Storage. The surgery must be pending and not refer to completed past surgeries.
- The patient(s) involved must carry tag(s) with QR codes.

4.4.4.5 Test Description

During the validation of this use case, the tester follows the steps below:

1. The testers signs in the MATISSE Mobile Application.
2. The tester selects the Planned Surgeries option.
3. The mobile app QR scanner is activated and the tester scans the patient's tag.
4. The tester visualizes the pending surgery item for the scanned patient, if any, through the mobile app.
5. The tester selects the surgery item and visualizes details about it to verify that the correct surgery process is applied to the current patient.

4.4.4.6 Input

The input of this test is considered to be the patient's tags that are going to be scanned as described in the previous section.

4.4.4.7 Test Setup

The MATISSE components involved in this test are the following:

- Data Storage (DS)
- Access Manager (AM)
- Session Manager (SM)
- Smart Tag Scanner (SSc)

4.4.4.8 Expected Result

Firstly, the tester signs in the MATISSE Mobile Application and selects the Planned Surgeries option. The QR scanner automatically launches and urges the tester to scan the patient's tag for a Pre-surgery Check session. The tester scans the patient's tag and, if existent, his/hers scheduled surgeries appear on the screen along with his/hers name and photo. The tester can visualize information about these scheduled surgeries.

4.4.4.9 Outcome

The tester signed in the MATISSE Mobile Application successfully and selected the Planned Surgeries option. The QR scanner launched automatically and the tester scanned a patient's tag for a Pre-surgery Check session (Figure 36). The patient's pending surgery appears on the screen along with his/hers name and photo. The tester can select the surgery and visualize its details (Figure 37).

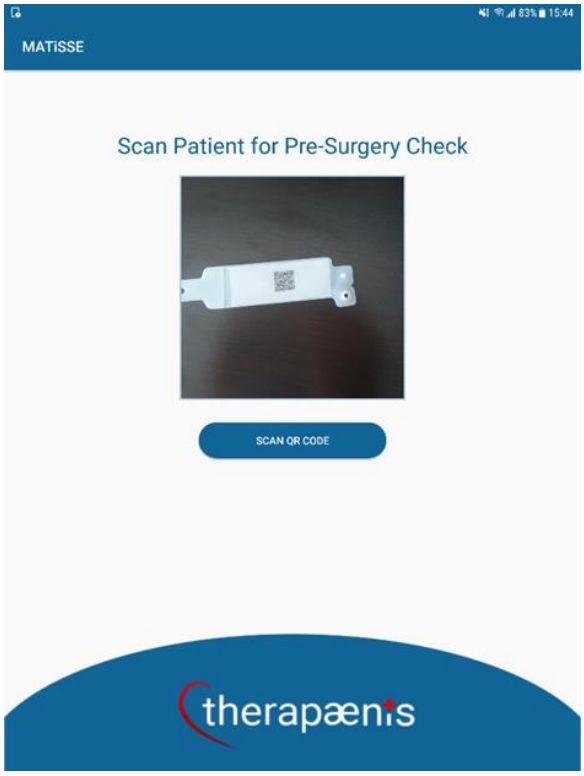


Figure 36: The QR scanner is launched for Pre-surgery Check

Figure 37: The pending surgery appears and the tester visualizes its details

4.4.4.10 Checkpoints

Checkpoint	Result	Comments
Is it possible to select the “Planned Surgeries” option?	PASS	The tester has already signed in with an Administrator account.
Is the QR scanner automatically activated?	PASS	After the selection of “Planned Surgeries” option.
Is the QR scanning possible?	PASS	After clicking the “Scan QR Code” button the scanning is enabled.
In case there are planned surgeries for the patient, is information provided?	PASS	The application fetches the scheduled future surgeries.
In case there are no planned surgeries for the patient, is the user notified accordingly?	PASS	The patient’s data are fetched but no surgery is displayed.

4.4.5 Patients’ Monitoring

4.4.5.1 Test Item

The item under test is the “Patients’ Monitoring” Use Case.

4.4.5.2 Objective

The test objectives is the verification of a Patient's Monitoring session, in which a clinician can scan a specific patient and fetch information about a scheduled surgery.

4.4.5.3 Features to be Tested

The following features are going to be tested:

- The login activity of the MATISSE Mobile Application.
- The operation of the Smart Tag Scanner (SSc) component.
- Scanning the patients' smart tags.
- Visualization of patients' and their medical history record on the mobile app screen.

4.4.5.4 Test Prerequisites

- An Android tablet or smart phone with the MATISSE mobile app installed must be available.
- The mobile device must have adequate battery resources at the beginning of the process.
- Communication between the MATISSE mobile app and the MATISSE backend must be available.
- The hospital worker (e. g. doctor) must have a valid account with proper access rights in the Patients' Monitoring use case.
- At least one patient and at least one associated prescription, exam, surgery, admission or allergy record must be stored in the MATISSE Data Storage.
- The patient(s) involved must carry tag(s) with QR codes.

4.4.5.5 Test Description

During the validation of this use case, the tester must follow the steps below:

1. The tester signs in the MATISSE mobile application.
2. The tester selects the Patients' Monitoring option.
3. The mobile app QR scanner is activated and the tester scans the patient's tag.
4. The tester visualizes the list of prescriptions, exams, surgeries, hospital admissions or allergies for the scanned patient, if any, through the mobile app.
5. The tester selects any item and visualizes details about it to get informed about the current patient's health record.

4.4.5.6 Input

The input of this test is considered to be the patient's tags that are going to be scanned as described in the previous section.

4.4.5.7 Test Setup

The MATISSE components involved in this test are the following:

- Data Storage (DS)
- Access Manager (AM)
- Session Manager (SM)
- Smart Tag Scanner (SSc)

4.4.5.8 Expected Result

Firstly, the tester signs in the MATISSE Mobile Application and selects the Patients' Monitoring option. The QR scanner automatically launches and urges the tester to scan the patient's tag for a Patient's Monitoring session. The tester scans the patient's tag and, if existent, his/hers hospital admissions, active and past prescriptions, exams, completed and scheduled surgeries and allergies are visualized in the mobile app screen by category.

4.4.5.9 Outcome

Firstly, the tester signs in the MATISSE Mobile Application and selects the Patients' Monitoring option. The QR scanner automatically launches and the tester scans the patient's tag for a Patient's Monitoring session (Figure 38). The patient's hospital admissions, active and past prescriptions, exams, completed and scheduled surgeries and allergies are visualized in the mobile app screen by category. (Figure 39 and Figure 40).

Figure 38: The QR scanner launches automatically for Patient's Monitoring

Figure 39: The testers enters the Patient's Monitoring session and visualizes a list of exams

Figure 40: The tester visualizes medical history information

4.4.5.10 Checkpoints

Checkpoint	Result	Comments
Is it possible to select the "Patients' Monitoring" option?	PASS	The tester has already signed in with an Administrator account.
Is the QR scanner automatically activated?	PASS	After the selection of "Patients' Monitoring" option.
Is the QR scanning possible?	PASS	After clicking the "Scan QR Code" button the scanning is enabled.
In case there is information for the patient, is this provided?	PASS	Admissions, Prescriptions, Exams, Surgeries and Allergies are fetched along with the patient data.
Is this information correctly displayed?	PASS	The patient's name and photo, along with categorized data are displayed.
In case there are no records for the patient, is the user notified accordingly?	PASS	The patient is fetched with empty records.

5. Conclusion

In this document, the functional evaluation testing overview, design and implementation of MATISSE are presented. Particular focus has been placed onto the use cases identified in deliverable D4.1 to ensure that the functional requirements identified in deliverable D2.1 are satisfied.

The architecture of the evaluation testbed was presented, followed by the generic approach that was used to correctly and in a structured way conduct the tests. The tests that were executed were related to both the technological validity of the implemented prototype (individually and as a whole) proving that the individual functionality of the various MATISSE components is sane and secure, as well as the contextual validity of the integrated MATISSE prototype, proving that the operational requirements of MATISSE under an orchestrated, successive component services invocation are satisfied, since all relevant tests were successful.

6. References

[1] MATISSE, "Deliverable D2.1: MATISSE system architecture and design," 2018.

[2] MATISSE, "Deliverable D3.1: MATISSE Prototype," 2018.

[3] MATISSE, "Deliverable D3.2: Selected binaries," 2018.

[4] MATISSE, "Deliverable D4.1: pilot set-up & Data Management Plan," 2018.

[5] Django REST framework, "Home," [Online]. Available: http://www.django-rest-framework.org.

[6] IETF, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," [Online]. Available: https://tools.ietf.org/html/rfc7231.

[7] Swagger.io, [Online]. Available: https://swagger.io/.

Annex A

In the following the source code for the combined MATISSE Drugs' Stocktaking and In-Hospital Medication use cases is presented.

```
"""Source code for the combined MATISSE Drugs' Stocktaking and In-Hospital Medication use cases"""
__copyright__ = "Therapaenis PC"
__version__ = "1.0"


from .tests import APITestCases, Patient, CartDrawer


class TestSessions(APITestCases):

    def test_sessions(self):
        # Tests for Unauthenticated Requests (401)
        self.test_client.credentials()
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + '/patients/').status_code, 401)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/').status_code, 401)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/' +
str(self.partial_prescription_1_id) + '/').status_code, 401)
        self.assertEqual(self.test_client.get('/sessions/medication_distribution/carts/'
+ str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/').status_code,
401)

        # Test Unauthorized Requests (403)
        self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' +
self.token_meddist.token)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + '/patients/').status_code, 403)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/').status_code, 403)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/' +
str(self.partial_prescription_1_id) + '/').status_code, 403)

        self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' +
self.token_medexams.token)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + '/patients/').status_code, 403)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/').status_code, 403)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/' +
str(self.partial_prescription_1_id) + '/').status_code, 403)

        self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' +
self.token_psurgery.token)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
```

```
str(self.smart_cart_1.id) + '/patients/').status_code, 403)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/').status_code, 403)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/' +
str(self.partial_prescription_1_id) + '/').status_code, 403)

        self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' +
self.token_clinician.token)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + '/patients/').status_code, 403)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/').status_code, 403)
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/' +
str(self.partial_prescription_1_id) + '/').status_code, 403)

        self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' +
self.token_phandler.token)
        self.assertEqual(self.test_client.get('/sessions/medication_distribution/carts/'
+ str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/').status_code,
403)

        self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' +
self.token_medexams.token)
        self.assertEqual(self.test_client.get('/sessions/medication_distribution/carts/'
+ str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/').status_code,
403)

        self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' +
self.token_psurgery.token)
        self.assertEqual(self.test_client.get('/sessions/medication_distribution/carts/'
+ str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/').status_code,
403)

        self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' +
self.token_clinician.token)
        self.assertEqual(self.test_client.get('/sessions/medication_distribution/carts/'
+ str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/').status_code,
403)


    def test_drugs_stocktaking_and_distribution_scenario(self):

        # Set Prescription Handler Credentials
        self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' +
self.token_phandler.token)

        # Associate patients to drawers and test association
        response = self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + '/patients/')
        self.assertEqual(response.status_code, 200)
        for patient in response.json():
            self.assertNotEqual(patient['cart_drawer'], [])
```

```
        # Get doses of patient
        self.assertEqual(self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/').status_code, 200)

        # Check if doses are marked as ready
        response = self.test_client.get('/sessions/drugs_stocktaking/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/' +
str(self.partial_prescription_1_id) + '/')
        self.assertEqual(response.status_code, 200)
        dose = response.json()
        self.assertTrue(dose['is_ready'])
        self.assertIsNotNone(dose['prescription_handler'])

        # Set Medication Distributor Credentials
        self.test_client.credentials(HTTP_AUTHORIZATION='Bearer ' +
self.token_meddist.token)

        # Check if doses are marked completed and cart_drawer dissassociation
        response = self.test_client.get('/sessions/medication_distribution/carts/' +
str(self.smart_cart_1.id) + self.patient_1_url + 'prescription_doses/')
        self.assertEqual(response.status_code, 200)
        for dose in response.json():
            self.assertTrue(dose['is_completed'])
            self.assertNotEqual(dose['medication_distributor'], [])
        response = self.test_client.get(self.patient_1_url)
        patient = response.json()
        self.assertEqual(patient['cart_drawer'], [])
```