NARRATE

Needs for Digital Recording and Documentation of Ecclesiastical Cultural Treasures in Monasteries and Temples

Project Information

Project Title:	Needs for Digital Recording and Documentation of Ecclesiastical Cultural Treasures in Monasteries and Temples
Programme/Action Type/Call:	ERASMUS+ / KA220-HED - Cooperation partnerships in higher education / 2022
Contract Number:	2022-1-EL01-KA220-HED-000089867
Start date:	29/12/2022
Duration in months:	24
Project Coordinator:	ARISTOTLE UNIVERSITY OF THESSALONIKI

The purpose of NARRATE project is to codify the actual recording and documentation needs for the ecclesiastical cultural treasures, through a systematic study of the users' needs.

Consortium partners

NARRATE

Needs for Digital Recording and Documentation of Ecclesiastical Cultural Treasures in Monasteries and Temples

ΙΔΡΥΜΑ ΚΡΑΤΙΚΩΝ ΥΠΟΤΡΟΦΙΩΝ ΙΚΥ

Co-funded by the European Union

Contract no: 2022-1-EL01-KA220-HED-000089867

Document Information

Title	R3.2 Open-source and Semantic-based Ecclesiastical Data Repository and Data Discovery Service and User Interface
Deliverable No.	R3.2
Version	1.0
Type	☒Report ☐Demonstrator ☐ORDP ☐Ethics ☐Other
Work Package	Work package n°3 - Implementation and User Evaluation of the NARRATE Framework
Work Package Leader	Co-lead by Cognitive UX GmbH (CUX) and Aristotle University of Thessaloniki (AUTH)
Issued by	Cognitive UX GmbH
Issued date	13/09/2024
Due date	20/09/2024
Dissemination Level	☒Public ☐Confidential

Main Authors

Name	Organization
Argyris Constantinides, Mario Belk	COGNITIVE UX GMBH
Stella Sylaiou, Konstantinos Evangelidis	INTERNATIONAL HELLENIC UNIVERSITY

Contributing Partners

Organization
ARISTOTLE UNIVERSITY OF THESSALONIKI
SOFIA UNIVERSITY ST KLEMENT OHRIDSKI
ASSOCIATION FOR THE PROTECTION OF CULTURAL HERITAGE (KMKD)

Co-funded by the European Union

Abbreviations

API	Application Programming Interface
CH	Cultural Heritage
CSS	Cascading Style Sheets
HTML	Hypertext Mark-up Language
WP	Work Package

Executive Summary

The EU Erasmus+ “NARRATE: Needs for Digital Recording and Documentation of Ecclesiastical Cultural Treasures in Monasteries and Temples” (2022-1-EL01-KA220-HED-000089867) aims at identifying and promoting the needs and priorities concerning ecclesiastical Cultural Heritage (CH) documentation.

The current study is being performed to codify the actual recording and documentation needs for the ecclesiastical cultural treasures, through a systematic study of the users’ needs. NARRATE reflects an emphasis on documenting ecclesiastic CH treasures in ways that will enable stakeholders to narrate their intertwined histories, functions, and spiritual importance throughout time.

This document summarizes and reports the activities and outcomes of *Activity 3.2: Development and Integration of an Ecclesiastical-centered Data Repository and Data Discovery Service*. This software report focuses on the development and integration of the ecclesiastical-centered data repository and data discovery service based on open-source tools and standard data exchange mechanisms.

Table of contents

List of figures

Co-funded by the European Union

Co-funded by the European Union

1. Introduction

The role of this document is to report the activities on the development and integration of the NARRATE ecclesiastical-centered data repository and data discovery service. This software report initially provides an overview of the implemented NARRATE framework solution, followed by the architectural design and the technology stack. Next, we provide a detailed description of the implemented functionalities and the realization of the interactive dashboard, which is based on the design mockups that were introduced at an earlier stage in the project. Finally, the appendices at the end of the report provide information on how to access the online NARRATE framework, as well as a detailed description of the application programming interfaces (APIs). The outcome of this report constitutes the basis for the summative evaluation user study of the NARRATE framework.

Co-funded by the European Union

Contract number: 2022-1-EL01-KA220-HED-000089867

2. Overview of the NARRATE Framework

This section provides an overview of the implemented NARRATE framework, illustrated in **Figure 1**. At a high-level, the NARRATE Framework consists of two main components: *i) NARRATE Interactive Dashboard*; and *ii) NARRATE Server*. The *NARRATE Interactive Dashboard* acts as the main source of end-users' interactions, allowing end-users to manage their accounts and the ecclesiastical treasures, as well as get access to the project's educational and dissemination material regarding the digital recording and documentation of ecclesiastical treasures. The NARRATE Server comprises a Web application that exposes the implemented APIs through which end-users or third-party services can interact with and exchange data.

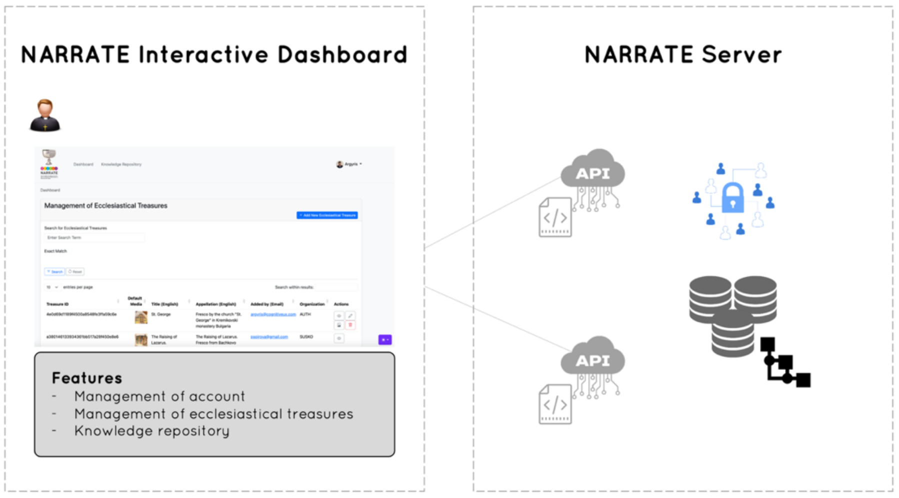

NARRATE Interactive Dashboard
NARRATE Server
Management of Ecclesiastical Treasures
Features
- Management of account
- Management of ecclesiastical treasures
- Knowledge repository
API
API

***Figure 1:** Overview of NARRATE Framework components.*

The realized solution was based on the conceptual design (**Figure 2**) described in the published report *R7 - Production of a conceptual framework that will identify the key characteristics, functionalities affordances, and modalities of the digital tools NARRATE will develop* [2]. Next, we describe the architectural design of the realized NARRATE framework, along with the technology stack and the implementation details.

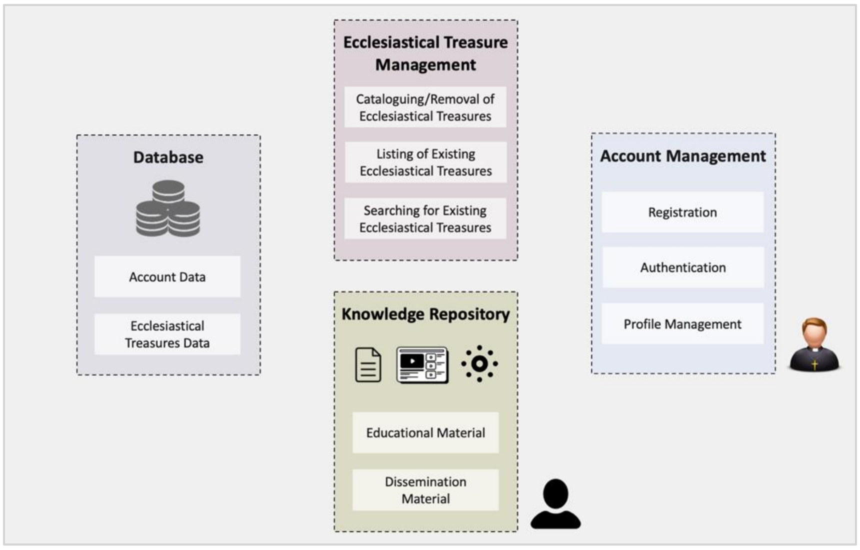

Ecclesiastical Treasure Management
Cataloguing/Removal of Ecclesiastical Treasures
Listing of Existing Ecclesiastical Treasures
Searching for Existing Ecclesiastical Treasures
Database
Account Data
Ecclesiastical Treasures Data
Account Management
Registration
Authentication
Profile Management
Knowledge Repository
Educational Material
Dissemination Material

Figure 2: High-level representation of the NARRATE conceptual framework.

3. Architectural Design and Technology Stack

This section describes the architecture of the NARRATE server-side Web API, which is illustrated in **Figure 3.** The architectural design has been introduced in the published report *R5 – Production of a framework for best practices guides* [1]. The server-side Web API is implemented in Python 3.10.8, using the Django REST Framework, which is an open-source Python and Django library used for building Web APIs. Furthermore, NGINX is utilized to deploy the server-side Web API. It acts as a versatile Web server, which could also be used as a reverse proxy and load balancer. Additionally, we employ the Gunicorn application server, which translates the HTTP requests into a format Python can process. Gunicorn implements the Web Server Gateway Interface, which is a standard interface between Web server software and Web applications.

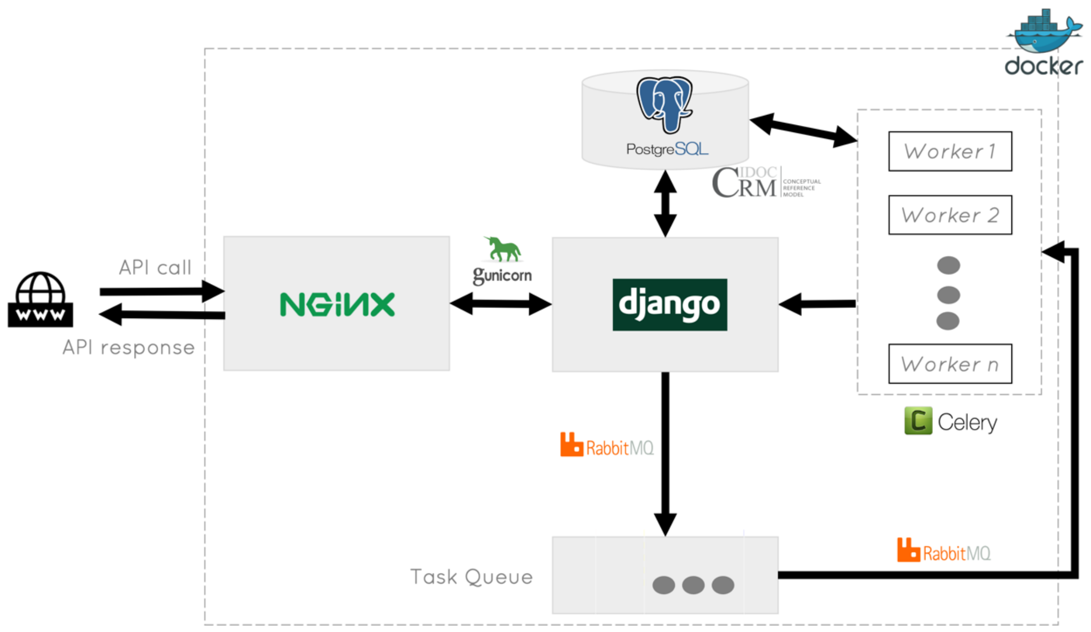


Figure 3: Architecture design of NARRATE Framework.

For certain time-consuming or blocking tasks (*e.g.*, sending emails with account verification during user registration), ideally we would like the request and response cycle to be fast. To address such time-consuming or blocking situations, we employ the Celery asynchronous task queue, which is based on distributed message passing. Celery requires an external solution for sending and receiving messages. For this purpose, we also use RabbitMQ, which is an open-source message-broker software that implements the Advanced Message Queuing Protocol.

For the storage of the data, PostgreSQL is used, which is an open-source relational database management system. The database tables are implemented by considering the CIDOC conceptual reference model, which is widely used for information integration in the area of cultural heritage. The Entity-Relationship diagram of the database is illustrated in **Figure 4**. Last, we use Docker platform for easily packing, shipping, and running our Web API as a lightweight, portable, and self-sufficient container.

Co-funded by the European Union

Contract number: 2022-1-EL01-KA220-HED-000089867

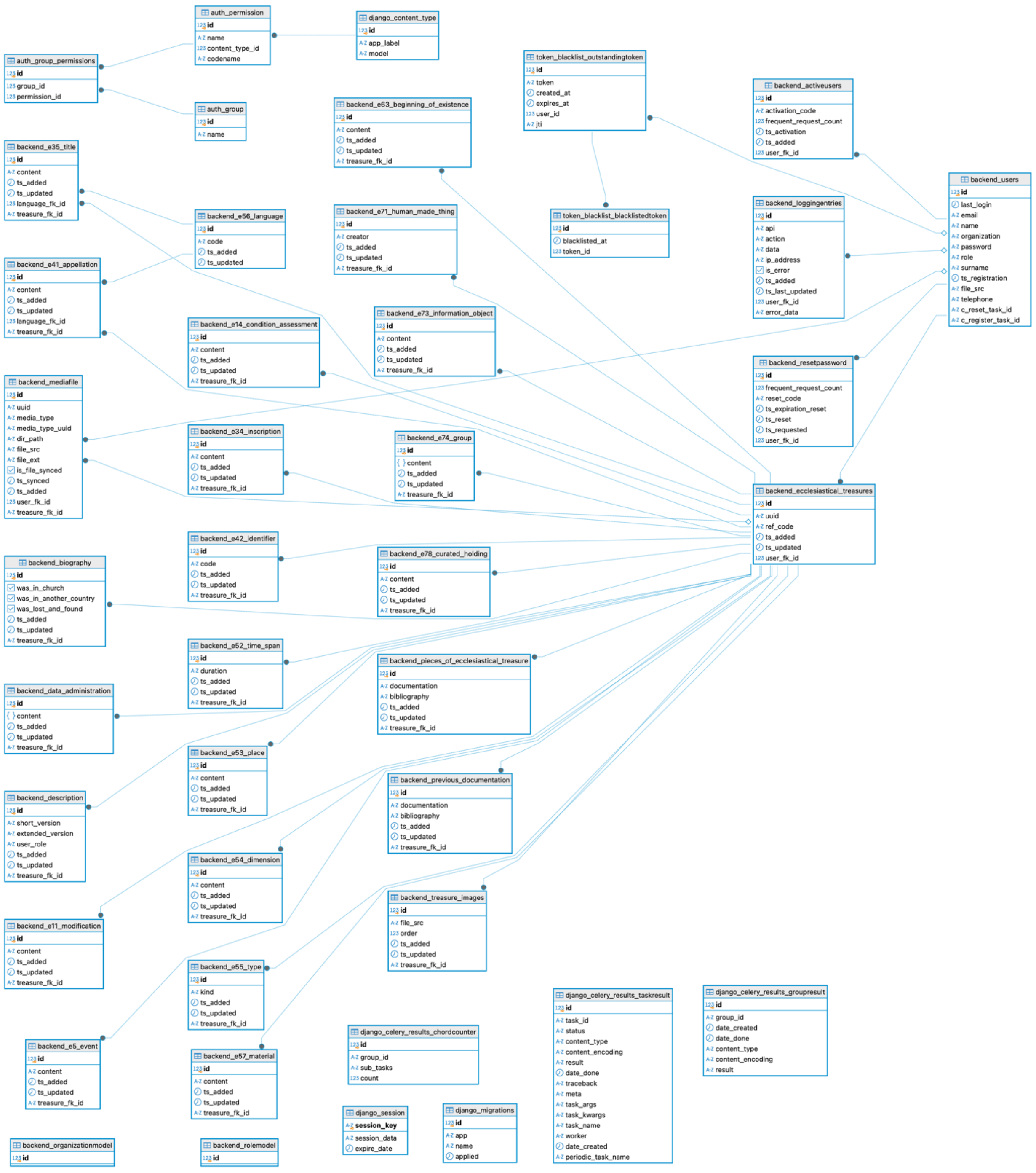

auth_permission: id, name, content_type_id, codename
django_content_type: id, app_label, model
auth_group_permissions: id, group_id, permission_id
auth_group: id, name
token_blacklist_outstandingtoken: id, token, created_at, expires_at, user_id, jti
backend_activeusers: id, activation_code, frequent_request_count, ts_activation, ts_added, user_fk_id
backend_e63_beginning_of_existence: id, content, ts_added, ts_updated, treasure_fk_id
backend_e35_title: id, content, ts_added, ts_updated, language_fk_id, treasure_fk_id
backend_e56_language: id, code, ts_added, ts_updated
backend_e71_human_made_thing: id, creator, ts_added, ts_updated, treasure_fk_id
token_blacklist_blacklistedtoken: id, blacklisted_at, token_id
backend_loggingentries: id, api, action, data, ip_address, is_error, ts_added, ts_last_updated, user_fk_id, error_data
backend_users: id, last_login, email, name, organization, password, role, surname, ts_registration, file_src, telephone, c_reset_task_id, c_register_task_id
backend_e41_appellation: id, content, ts_added, ts_updated, language_fk_id, treasure_fk_id
backend_e14_condition_assessment: id, content, ts_added, ts_updated, treasure_fk_id
backend_e73_information_object: id, content, ts_added, ts_updated, treasure_fk_id
backend_resetpassword: id, frequent_request_count, reset_code, ts_expiration_reset, ts_reset, ts_requested, user_fk_id
backend_mediafile: id, uuid, media_type, media_type_uuid, dir_path, file_src, file_ext, is_file_synced, ts_synced, ts_added, user_fk_id, treasure_fk_id
backend_e34_inscription: id, content, ts_added, ts_updated, treasure_fk_id
backend_e74_group: id, content, ts_added, ts_updated, treasure_fk_id
backend_ecclesiastical_treasures: id, uuid, ref_code, ts_added, ts_updated, user_fk_id
backend_e42_identifier: id, code, ts_added, ts_updated, treasure_fk_id
backend_e78_curated_holding: id, content, ts_added, ts_updated, treasure_fk_id
backend_biography: id, was_in_church, was_in_another_country, was_lost_and_found, ts_added, ts_updated, treasure_fk_id
backend_e52_time_span: id, duration, ts_added, ts_updated, treasure_fk_id
backend_pieces_of_ecclesiastical_treasure: id, documentation, bibliography, ts_added, ts_updated, treasure_fk_id
backend_data_administration: id, content, ts_added, ts_updated, treasure_fk_id
backend_e53_place: id, content, ts_added, ts_updated, treasure_fk_id
backend_previous_documentation: id, documentation, bibliography, ts_added, ts_updated, treasure_fk_id
backend_description: id, short_version, extended_version, user_role, ts_added, ts_updated, treasure_fk_id
backend_e54_dimension: id, content, ts_added, ts_updated, treasure_fk_id
backend_treasure_images: id, file_src, order, ts_added, ts_updated, treasure_fk_id
backend_e11_modification: id, content, ts_added, ts_updated, treasure_fk_id
backend_e55_type: id, kind, ts_added, ts_updated, treasure_fk_id
django_celery_results_taskresult: id, task_id, status, content_type, content_encoding, result, date_done, traceback, meta, task_args, task_kwargs, task_name, worker, date_created, periodic_task_name
django_celery_results_groupresult: id, group_id, date_created, date_done, content_type, content_encoding, result
django_celery_results_chordcounter: id, group_id, sub_tasks, count
backend_e5_event: id, content, ts_added, ts_updated, treasure_fk_id
backend_e57_material: id, content, ts_added, ts_updated, treasure_fk_id
django_session: session_key, session_data, expire_date
django_migrations: id, app, name, applied
backend_organizationmodel: id
backend_rolemodel: id

Figure 4: Entity-Relationship datagram of the database.

The NARRATE Interactive Dashboard front-end is developed using the Django's template language, which contains the following: *i)* variables, which take values upon rendering of the Hypertext Mark-up Language (HTML) template file; and *ii)* tags, which

Contract number: 2022-1-EL01-KA220-HED-000089867

control the logic in the rendering process. HTML is the primary language for creating web pages, providing a means for describing the structure of text information in a document. It defines the content and structure of web content, and is often used in combination with other technologies, such as, Cascading Style Sheets (CSS) which describe the appearance of the web page, and JavaScript that controls the functionality and behavior of the web page. The NARRATE Interactive Dashboard front-end was implemented utilizing the latest version of HTML5 to leverage its enhanced features and ensure compatibility with modern standards of web browsers. With regards to the styling of the NARRATE Interactive Dashboard front-end, we utilized the latest version of CSS to leverage advanced styling features and improvements for enhanced web presentation, while conforming to the latest design standards of the World Wide Web Consortium. To handle the interactions of the NARRATE Interactive Dashboard front-end, we utilized JavaScript, which is the most commonly used client-side scripting language. JavaScript is used to handle end-users' interactions, as well as for communication and exchange of data with the NARRATE server in an asynchronous manner (*i.e.*, without the need for reloading the web page).

Co-funded by the European Union

Contract number: 2022-1-EL01-KA220-HED-000089867

4. Interactive Dashboard

This section presents the realization of the NARRATE Interactive Dashboard, which is based on the design mockups that have been introduced in the published report *R7* [2]. Next, we describe the implemented functionalities in each page of the NARRATE Interactive Dashboard. The Uniform Resource Locators (URLs) for accessing the NARRATE Framework can be found in the *APPENDIX A – Access URLs to the NARRATE Framework*, while the complete list of the implemented APIs can be found in the *APPENDIX B – Application Programming Interfaces*.

4.1 Account Management

This section describes the implementation details for the management of end-users accounts, which follows state-of-the-art approaches in user authentication, and includes the following functionalities: *i)* sign up to the NARRATE framework using a valid email address; *ii)* account activation via an activation code received in the email address; *iii)* reset password via a reset code received in the email address; *iv)* update of basic profile details; and *v)* update of password.

The NARRATE repository is hosted by Cognitive UX GmbH and is available at the following link:

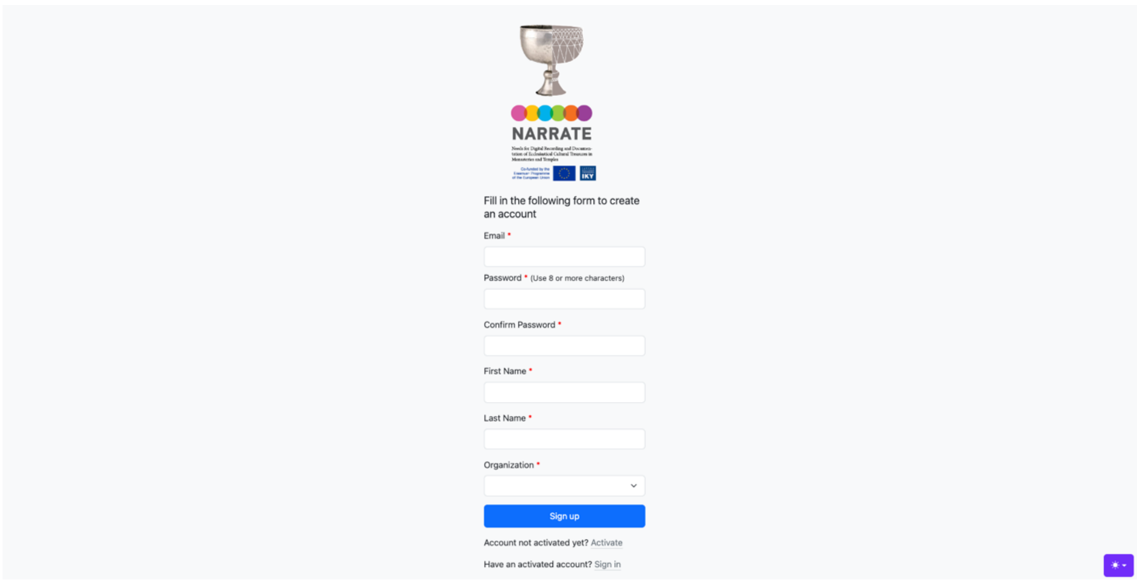
Login :: NARRATE

4.1.1 Sign up

The sign up page (**Figure 5**) of the NARRATE Interactive Dashboard allows individuals to create an end-user account in order to get access to the platform. The individual is requested to fill in the required fields (*i.e.*, email, password, confirm password, first name, last name, organization) in the sign up form. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen. Upon successful completion of the action, an activation email is sent to the provided email address. The activation email includes an activation code which is required for activating the end-user account through the activate account page.

Co-funded by the European Union

Contract number: 2022-1-EL01-KA220-HED-000089867



Fill in the following form to create an account

Email *

Password * (Use 8 or more characters)

Confirm Password *

First Name *

Last Name *

Organization *

Sign up

Account not activated yet? Activate

Have an activated account? Sign in

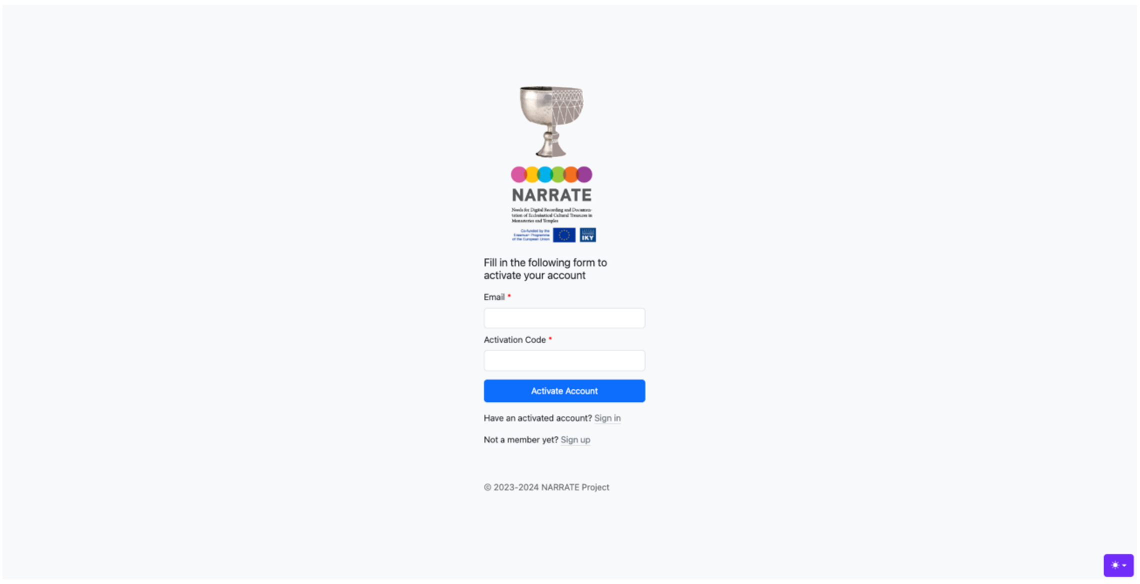
Figure 5: Sign up page allows individuals to create an end-user account.

4.1.2 Activate account

The activate account page (**Figure 6**) of the NARRATE Interactive Dashboard allows individuals to activate their account prior to getting access to the platform. The individual is requested to fill in the required fields (*i.e.*, email, activation code received in the email address) in the activate account form. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen. Upon successful completion of the action, the end-users's account gets activated and the end-user can access the platform through the sign in page.

Co-funded by the European Union

Contract number: 2022-1-EL01-KA220-HED-000089867



Fill in the following form to activate your account

Email *

Activation Code *

Activate Account

Have an activated account? Sign in

Not a member yet? Sign up

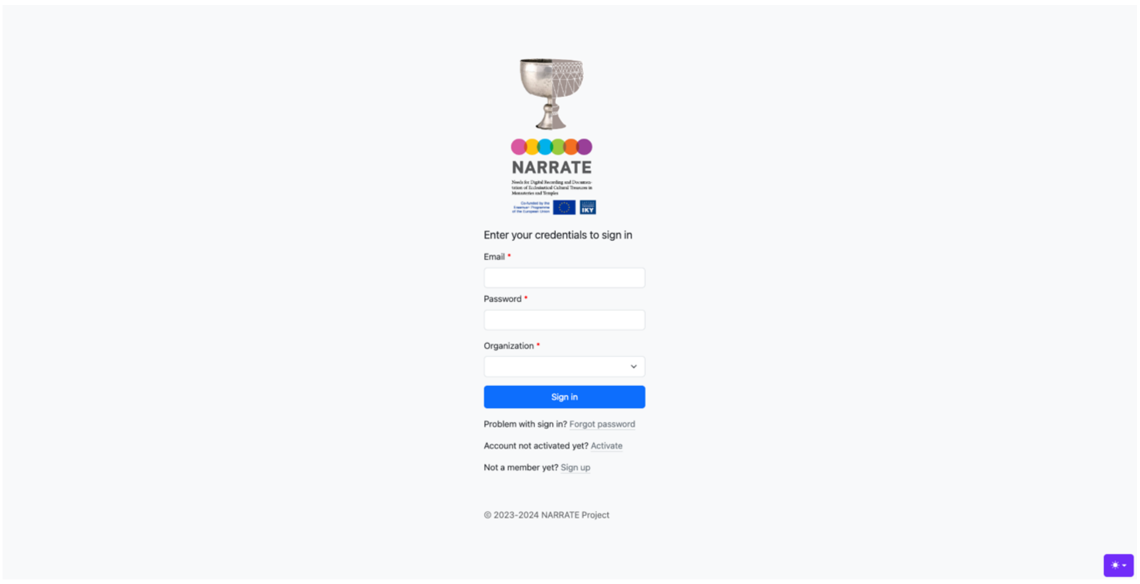
© 2023-2024 NARRATE Project

Figure 6: Activate account page allows individuals to activate their account prior to getting access to the platform.

4.1.3 Sign in

The sign in page (**Figure 7**) of the NARRATE Interactive Dashboard allows individuals to get access to the functionalities of the platform and the catalogued ecclesiastical treasures. The individual is requested to fill in the required fields (*i.e.*, email, password, organization) in the sign in form. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen. Upon successful completion of the action, the end-user can access the functionalities regarding the management of ecclesiastical treasures and the material found in the knowledge repository.

Co-funded by the European Union



NARRATE

Enter your credentials to sign in

Email *

Password *

Organization *

Sign in

Problem with sign in? Forgot password

Account not activated yet? Activate

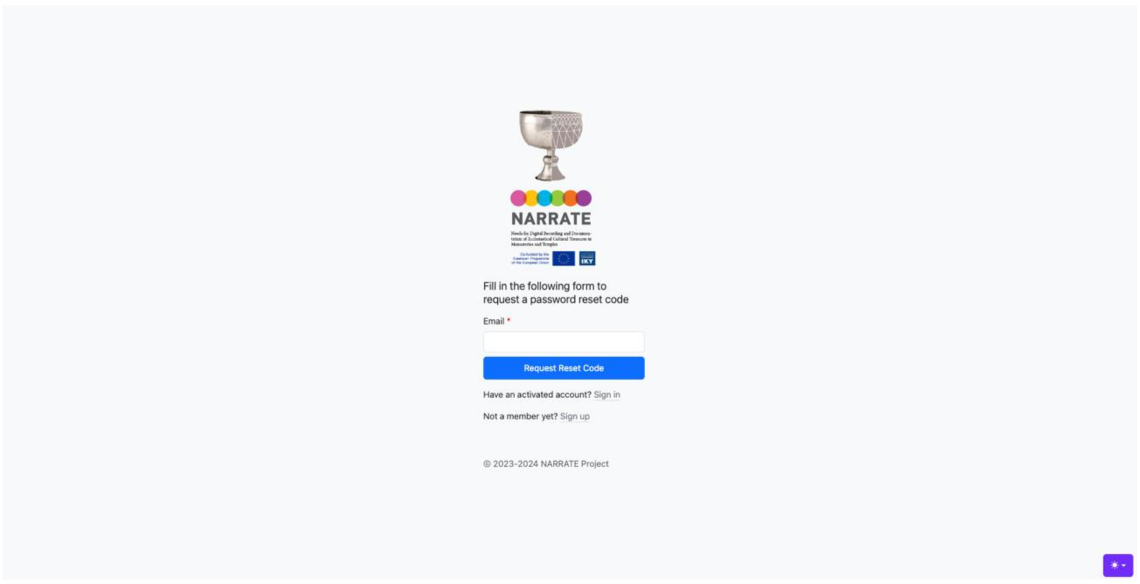
Not a member yet? Sign up

© 2023-2024 NARRATE Project

Figure 7: Sign in page allows individuals to get access to the functionalities of the platform and the catalogued ecclesiastical treasures.

4.1.4 Forgot password

The forgot password page (**Figure 8**) of the NARRATE Interactive Dashboard allows individuals to request a password reset code via email in cases they forgot their password and are not able to access the platform. The individual is requested to fill in the required field (*i.e.*, email) in the forgot password form. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen. Upon successful completion of the action, a password reset email is sent to the provided email address. The password reset email includes a reset code which is required for resetting the end-user's password through the reset password page.


NARRATE
Needs for Digital Recording and Documentation of Ecclesiastical Cultural Treasures in Monasteries and Temples

Fill in the following form to request a password reset code

Email *

Request Reset Code

Have an activated account? Sign in

Not a member yet? Sign up

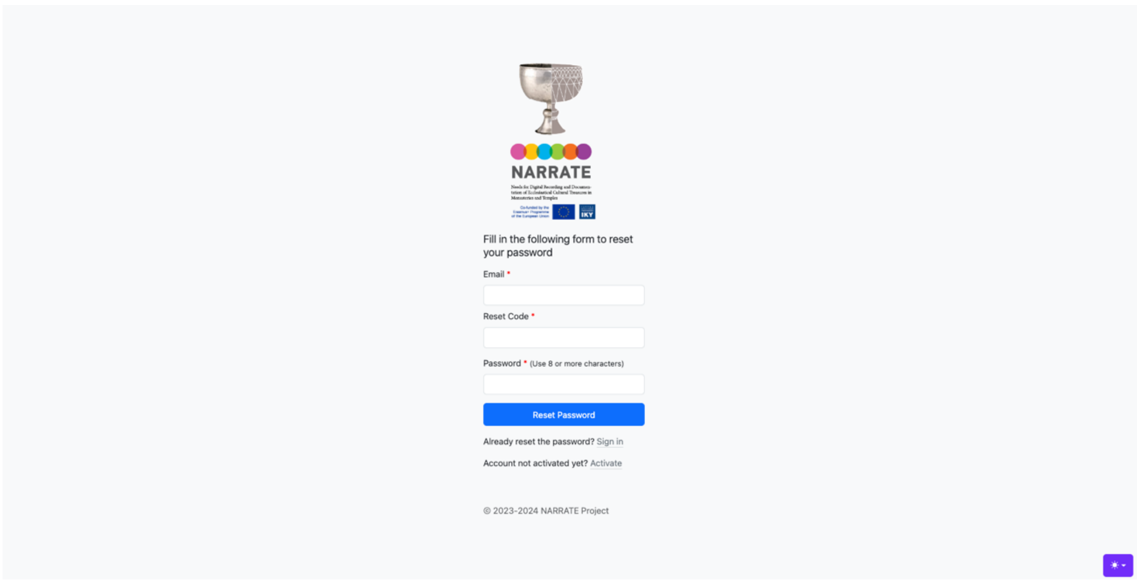
© 2023-2024 NARRATE Project

Figure 8: Forgot password page allows individuals to request a password reset code via email.

4.1.5 Reset password

The reset password page (**Figure 9**) of the NARRATE Interactive Dashboard allows individuals to reset their password by using the reset code received via email by clicking on an expiring password reset URL (**Figure 10**). The individual is requested to fill in the required fields (*i.e.*, email, reset code received in the email address, password) in the reset password form. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen. Upon successful completion of the action, the password gets reset and the end-user can sign in through the sign in page.

Co-funded by the European Union


NARRATE

Fill in the following form to reset your password

Email *

Reset Code *

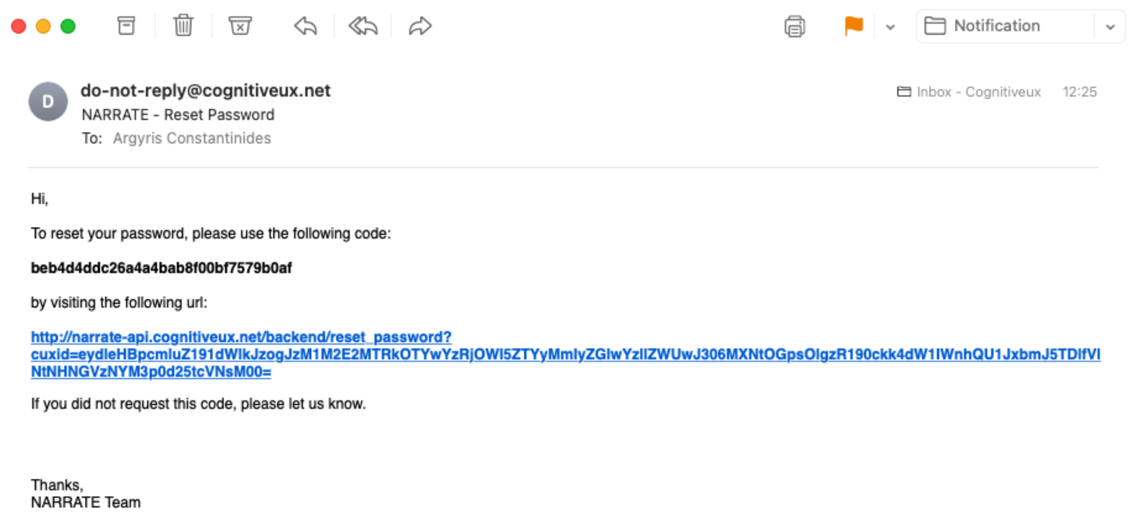
Password * (Use 8 or more characters)

Already reset the password? Sign in

Account not activated yet? Activate

© 2023-2024 NARRATE Project

Figure 9: Reset password page allows individuals to reset their password.



do-not-reply@cognitiveux.net
NARRATE - Reset Password
To: Argyris Constantinides

Inbox - Cognitiveux 12:25

Hi,

To reset your password, please use the following code:

beb4d4ddc26a4a4bab8f00bf7579b0af

by visiting the following url:

http://narrate-api.cognitiveux.net/backend/reset_password?cuxid=eydleHBpcmluZ191dWlkJzogJzM1M2E2MTRkOTYwYzRjOWI5ZTYyMmlyZGlwYzllZWUwJ306MXNtOGpsOlgzR190ckk4dW1lWnhQU1JxbmJ5TDlfVlNtNHNGVzNYM3p0d25tcVNsM00=

If you did not request this code, please let us know.

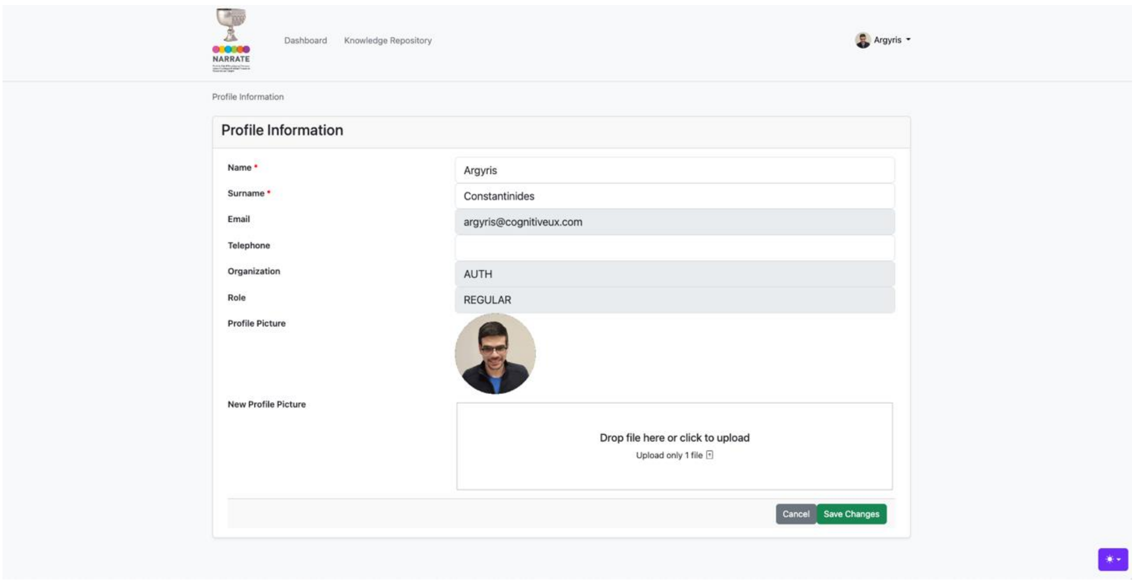
Thanks,
NARRATE Team

Figure 10: End-users receive the reset code via email and they should click on th expiring password reset URL in order to access the password reset page.

4.1.6 Update profile details (requires Sign in)

The update profile page (**Figure 11**) of the NARRATE Interactive Dashboard allows individuals to update their profile information. The individual is requested to fill in the required fields (*i.e.*, name, surname), as well as any of the non-required fields (*i.e.*, telephone, new profile picture), in the update profile details form. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen.

Upon successful completion of the action, the profile details of the end-user are updated. This functionality is available by clicking at the top right of the page (*i.e.*, where the name of the signed in user is displayed), and then by selecting the option “My Profile”.



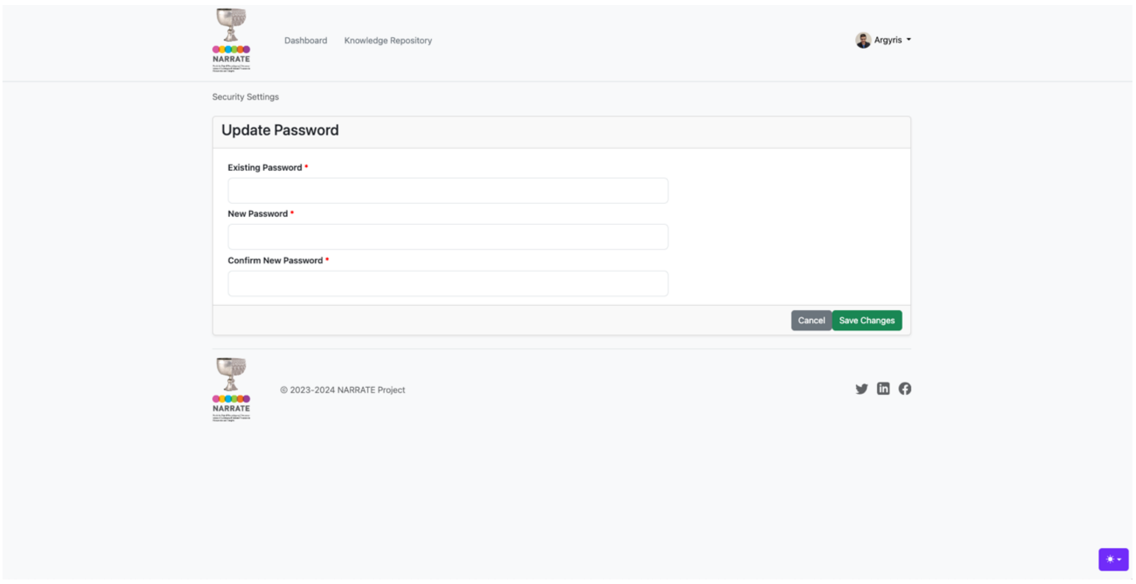
Profile Information

Name *	Argyris
Surname *	Constantinides
Email	argyris@cognitiveux.com
Telephone	
Organization	AUTH
Role	REGULAR
Profile Picture	
New Profile Picture	Drop file here or click to upload Upload only 1 file

Figure 11: Update profile page allows individuals to update their profile information.

4.1.7 Update password (requires Sign in)

The update security settings page (**Figure 12**) of the NARRATE Interactive Dashboard allows individuals to update their password. The individual is requested to fill in the required fields (*i.e.*, existing password, new password, confirm new password) in the update password form. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen. Upon successful completion of the action, the password gets updated and the end-user can sign in to the system using the updated password. This functionality is available by clicking at the top right of the page (*i.e.*, where the name of the signed in user is displayed), and then by selecting the option “Security Settings”.



Update Password

Existing Password *

New Password *

Confirm New Password *

© 2023-2024 NARRATE Project

Figure 12: Update security settings page allows individuals to update their password.

4.1.8 Sign out (requires Sign in)

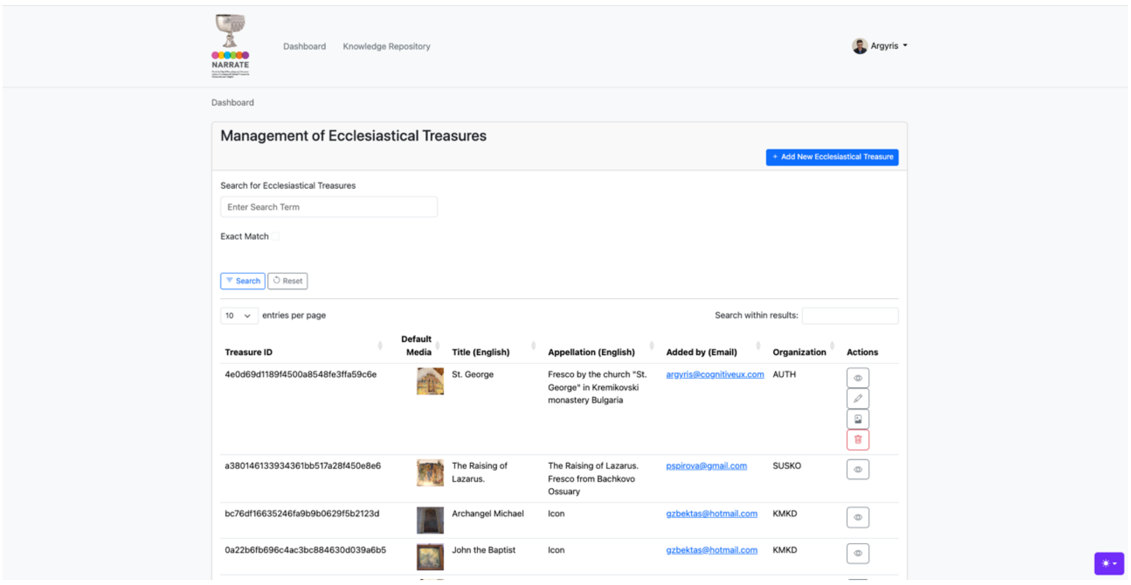
The sign out functionality of the NARRATE Interactive Dashboard allows individuals to terminate their signed in session and sign out of the system. This functionality is available by clicking at the top right of the page (*i.e.*, where the name of the signed in user is displayed), and then by selecting the option “Sign out”.

4.2 Ecclesiastical Treasures Management

This section describes the implementation details for the management of the ecclesiastical treasures, which includes the following functionalities: *i)* View existing ecclesiastical treasures; *ii)* Add new ecclesiastical treasure; *iii)* Update existing ecclesiastical treasure; *iv)* Delete existing ecclesiastical treasure; *v)* Advanced search (*e.g.*, free text search, exact match search); and *vi)* Basic access control (*i.e.*, end-users can see all the ecclesiastical treasures but can update/delete only the ecclesiastical treasures added by them). The functionalities *i)* – *iv)* and *vi)* include both the descriptive details and the media files of the ecclesiastical treasures.

4.2.1 View existing ecclesiastical treasures (requires Sign in)

The view existing ecclesiastical treasures page (**Figure 13**) of the NARRATE Interactive Dashboard is the landing page in which the end-user is redirected after sign in to the platform. This page displays the basic information of all the catalogued ecclesiastical treasures in a tabular format. For each ecclesiastical treasure listed in the table, there are available actions based on the access control. For instance, for the ecclesiastical treasures added by the signed in user, there are four available actions as follows: *i)* view the descriptive details and the media files of the ecclesiastical treasure; *ii)* update the descriptive details of the ecclesiastical treasure; *iii)* manage the media files (*i.e.*, view/add/update/delete) of the ecclesiastical treasure; and *iv)* delete the ecclesiastical treasure. For the ecclesiastical treasures added by other end-users (*i.e.*, not the signed in user), there is only one available action, which is viewing the descriptive details and the media files of the ecclesiastical treasure. The view existing ecclesiastical treasures page also contains the “Add New Ecclesiastical Treasure” button, which redirects end-users to another screen that allows them to catalogue a new ecclesiastical treasure.



Management of Ecclesiastical Treasures

Search for Ecclesiastical Treasures

Enter Search Term

Exact Match

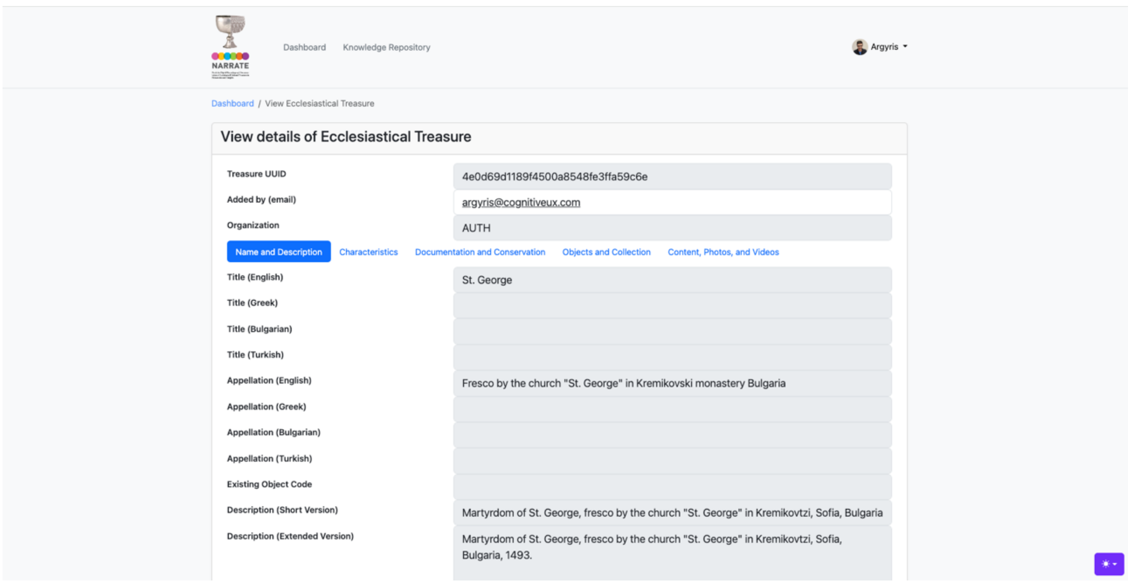
10 entries per page

Search within results:

Treasure ID	Default Media	Title (English)	Appellation (English)	Added by (Email)	Organization	Actions
4e0d69d1189f4500a8548fe3ffa59c6e		St. George	Fresco by the church "St. George" in Kremikovski monastery Bulgaria	argyris@cognitiveux.com	AUTH	
a380146133934361bb517a28f450e8e6		The Raising of Lazarus.	The Raising of Lazarus. Fresco from Bachkovo Ossuary	pspirova@gmail.com	SUSKO	
bc76df16635246fa9b9b0629f5b2123d		Archangel Michael	Icon	gzbektas@hotmail.com	KMKD	
0a22b6fb696c4ac3bc884630d039a6b5		John the Baptist	Icon	gzbektas@hotmail.com	KMKD	

Figure 13: View existing ecclesiastical treasures page displays the basic information of all the catalogued ecclestical treasures.

Upon clicking the eye icon in the Actions area for a specific ecclesiastical treasure, the end-user is redirected to another screen that displays all the descriptive details and the media files of the ecclesiastical treasure, as illustrated in **Figure 14**.



Dashboard Knowledge Repository Argyris

Dashboard / View Ecclesiastical Treasure

View details of Ecclesiastical Treasure

Treasure UUID	4e0d69d1189f4500a8548fe3ffa59c6e
Added by (email)	argyris@cognitiveux.com
Organization	AUTH

Name and Description | Characteristics | Documentation and Conservation | Objects and Collection | Content, Photos, and Videos

Title (English)	St. George
Title (Greek)	
Title (Bulgarian)	
Title (Turkish)	
Appellation (English)	Fresco by the church "St. George" in Kremikovski monastery Bulgaria
Appellation (Greek)	
Appellation (Bulgarian)	
Appellation (Turkish)	
Existing Object Code	
Description (Short Version)	Martyrdom of St. George, fresco by the church "St. George" in Kremikovtzi, Sofia, Bulgaria
Description (Extended Version)	Martyrdom of St. George, fresco by the church "St. George" in Kremikovtzi, Sofia, Bulgaria, 1493.

Figure 14: View-only page that displays all the descriptive details and the media files of the ecclesiastical treasure.

4.2.2 Add new ecclesiastical treasure (requires Sign in)

The add new ecclesiastical treasure page (**Figure 15, Figure 16**) of the NARRATE Interactive Dashboard allows individuals to catalogue a new ecclesiastical treasure. This is conducted through a series of steps, which are grouped at higher levels of abstraction, such as, “Name and Description”, “Characteristics”, “Documentation and Conservation”, “Objects and Collection”, and “Content, Photos, and Videos”.

Co-funded by the European Union

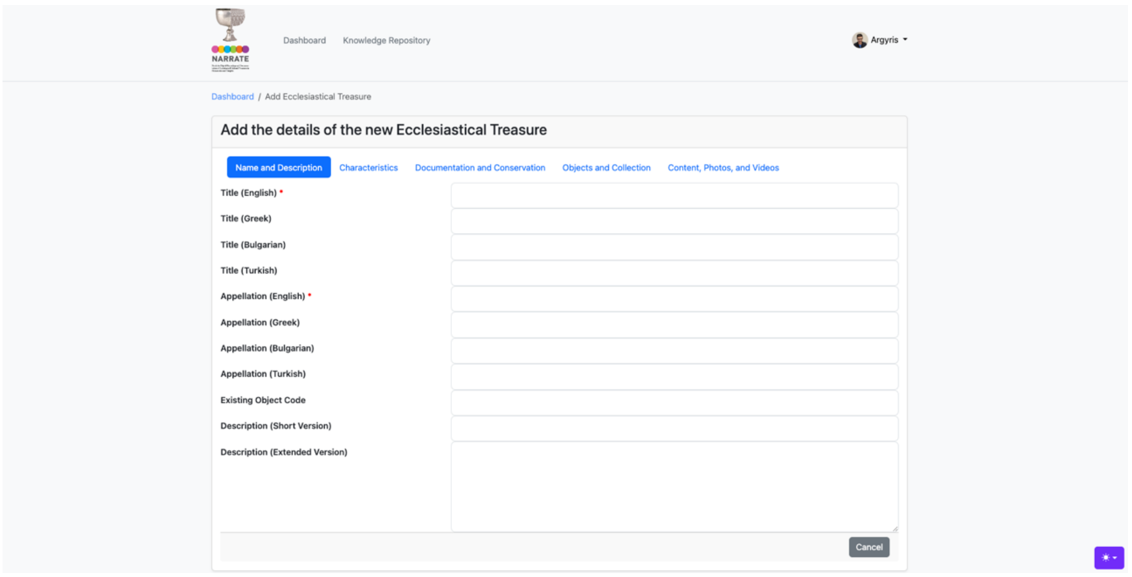


Figure 15: Add new ecclesiastical treasure page allows individuals to catalogue the details of a new ecclesiastical treasure.

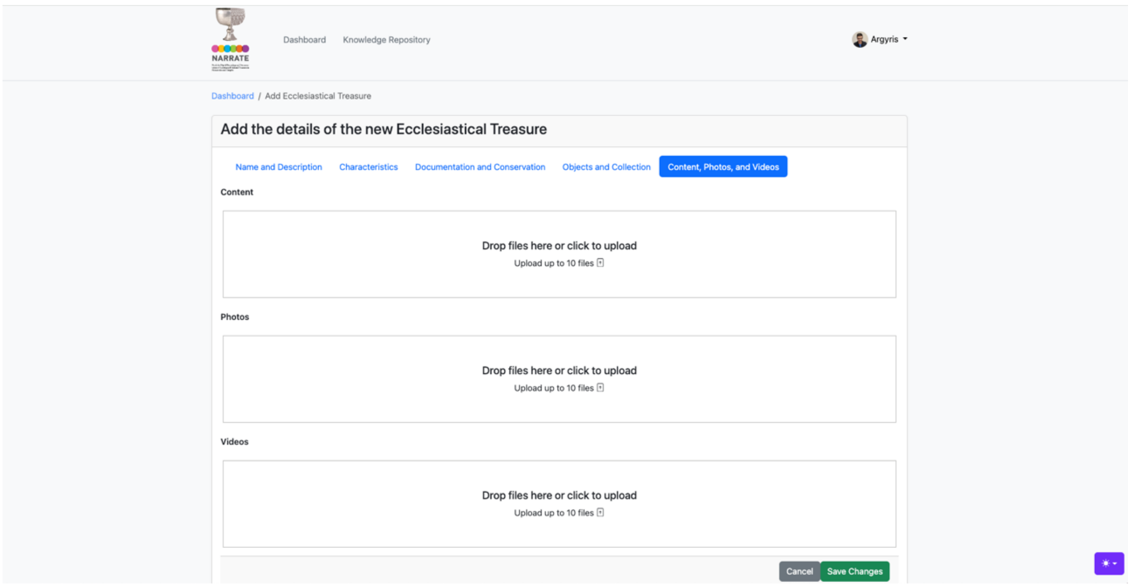
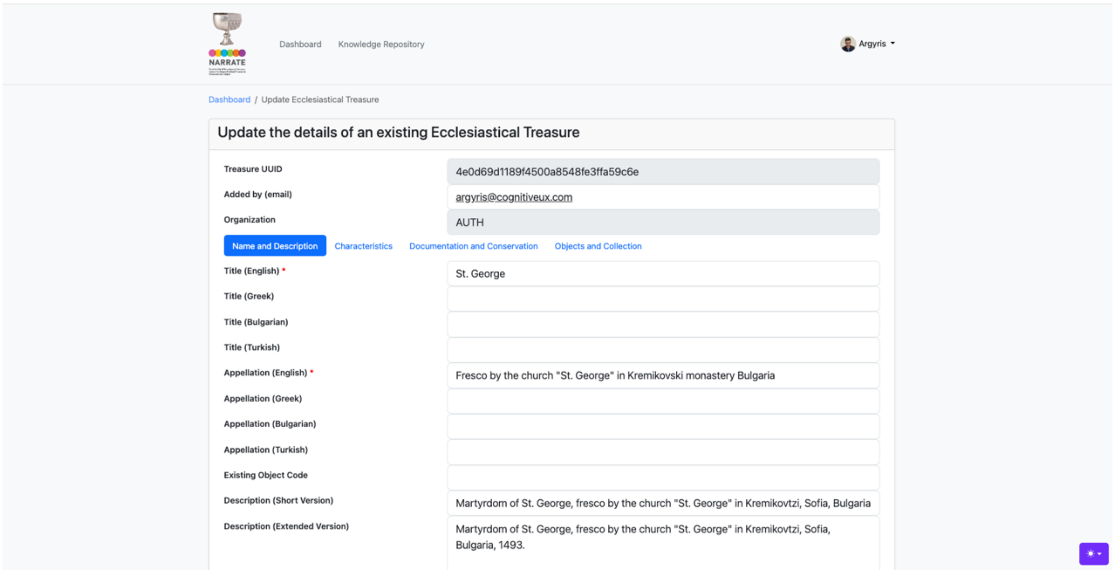

NARRATE
Dashboard
Knowledge Repository
Argyris
Dashboard / Add Ecclesiastical Treasure
Add the details of the new Ecclesiastical Treasure
Name and Description
Characteristics
Documentation and Conservation
Objects and Collection
Content, Photos, and Videos
Content
Drop files here or click to upload
Upload up to 10 files
Photos
Drop files here or click to upload
Upload up to 10 files
Videos
Drop files here or click to upload
Upload up to 10 files
Cancel
Save Changes

Figure 16: Add new ecclesiastical treasure page also allows individuals to catalogue the media files associated with the new ecclesiastical treasure.

The individual is requested to fill in the required fields (*i.e.*, title in English, appellation in English) in the add new ecclesiastical treasure form. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen. Upon successful completion of the action, the descriptive details and any media associated with the ecclesiastical treasure are catalogued in the platform.

4.2.3 Update existing ecclesiastical treasure (requires Sign in)

The update existing ecclesiastical treasure page (**Figure 17**) of the NARRATE Interactive Dashboard allows individuals to update the descriptive details of the ecclesiastical treasure. To reach this page, an end-user must first visit the view existing ecclesiastical treasures page (**Figure 13**) and click the pencil icon under the “Actions” of the respective treasure they wish to update. Then, the individual is requested to fill in the required fields (*i.e.*, title in English, appellation in English) in the update existing ecclesiastical treasure form. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen. Upon successful completion of the action, the descriptive details of the ecclesiastical treasure are updated in the platform.



Update the details of an existing Ecclesiastical Treasure

Treasure UUID	4e0d69d1189f4500a8548fe3ffa59c6e
Added by (email)	argyris@cognitiveux.com
Organization	AUTH
Title (English) *	St. George
Title (Greek)	
Title (Bulgarian)	
Title (Turkish)	
Appellation (English) *	Fresco by the church "St. George" in Kremikovski monastery Bulgaria
Appellation (Greek)	
Appellation (Bulgarian)	
Appellation (Turkish)	
Existing Object Code	
Description (Short Version)	Martyrdom of St. George, fresco by the church "St. George" in Kremikovtzi, Sofia, Bulgaria
Description (Extended Version)	Martyrdom of St. George, fresco by the church "St. George" in Kremikovtzi, Sofia, Bulgaria, 1493.

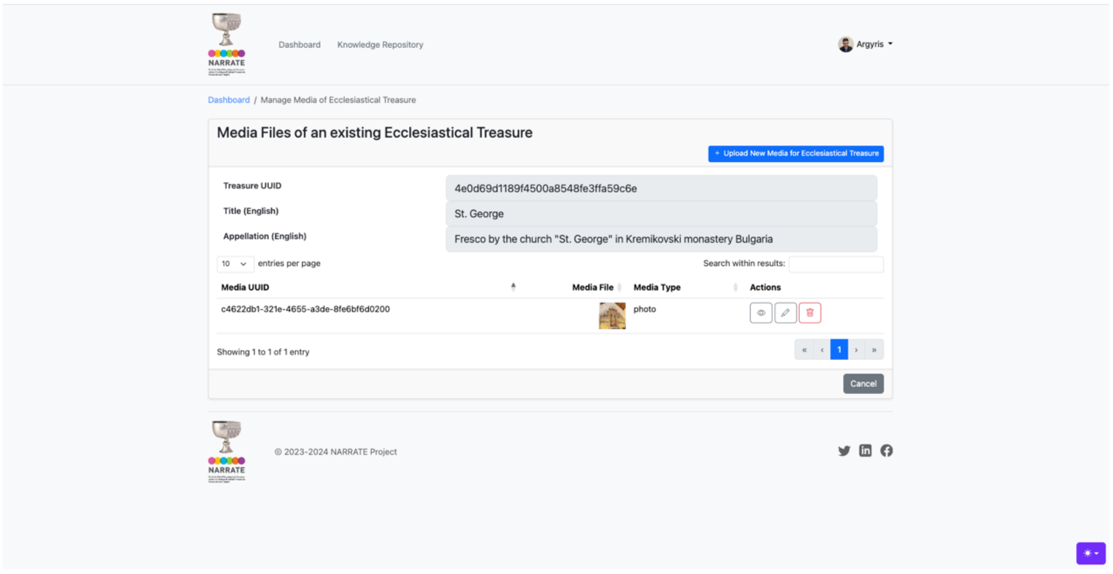
Figure 17: Update existing ecclesiastical treasure page allows individuals to update the descriptive details of the ecclesiastical treasure.

Regarding the management of the media of an ecclesiastical treasure, an end-user must first visit the view existing ecclesiastical treasures page (**Figure 13**) and click the media icon under the “Actions” of the respective treasure for which they wish to manage its media files. The end-user is then redirected to the media management page (**Figure 18**), through which they can perform the following actions: *i)* view directly a thumbnail of the media associated with the ecclesiastical treasure or view it in its original size by clicking

Co-funded by the European Union

Contract number: 2022-1-EL01-KA220-HED-000089867

the eye icon under the “Actions”; *ii)* Update the media by clicking the pencil icon under the “Actions”; and *iii)* delete the media by clicking the bin icon under the “Actions”. By selecting action *ii)*, the end-user will be redirected to the update media page (**Figure 19**), through which they are requested to fill in the required fields (*i.e.*, new media file) in the update media form. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen. Upon successful completion of the action, the media gets updated. By selecting action *iii)*, the end-user will be redirected to the delete media page (**Figure 20**), through which they are requested to review and confirm the deletion of the media. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen. Upon successful completion of the action, the media gets permanently deleted.



Dashboard / Manage Media of Ecclesiastical Treasure

Media Files of an existing Ecclesiastical Treasure

+ Upload New Media for Ecclesiastical Treasure

Treasure UUID	4e0d69d1189f4500a8548fe3ffa59c6e
Title (English)	St. George
Appellation (English)	Fresco by the church "St. George" in Kremikovski monastery Bulgaria

10 entries per page — Search within results:

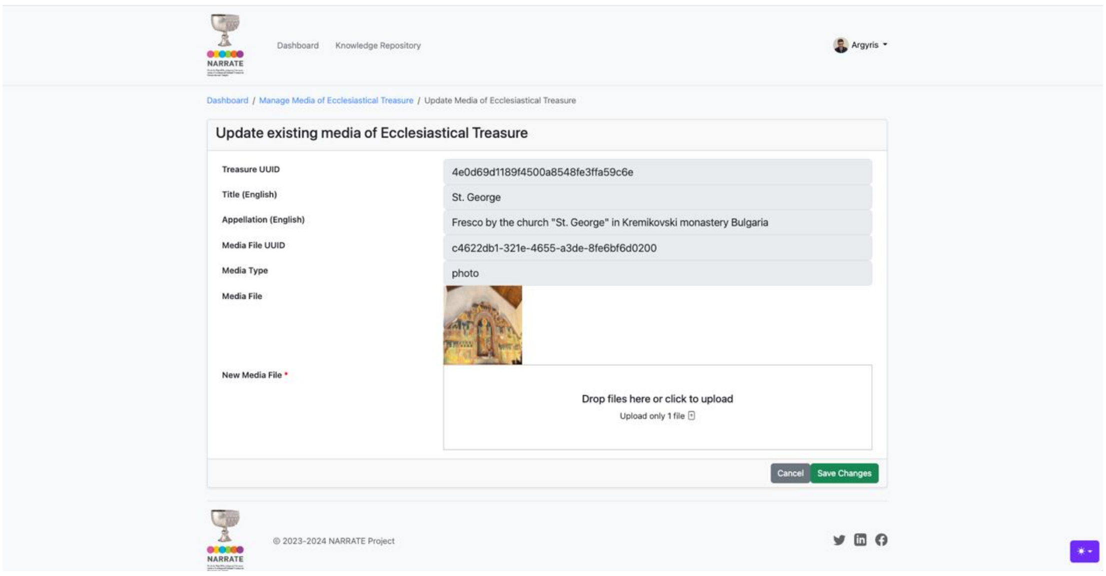
Media UUID	Media File	Media Type	Actions
c4622db1-321e-4655-a3de-8fe6bf6d0200		photo	

Showing 1 to 1 of 1 entry

Cancel

© 2023-2024 NARRATE Project

Figure 18: *Media management page through which end-users can view/update/delete the media associated with the ecclesiastical treasure.*

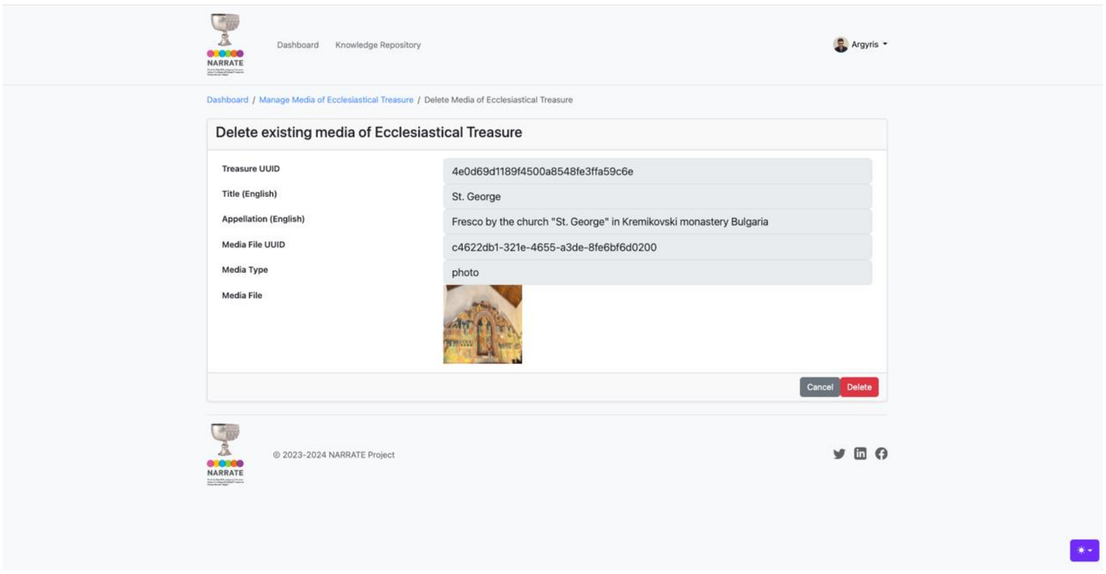


Dashboard / Manage Media of Ecclesiastical Treasure / Update Media of Ecclesiastical Treasure

Update existing media of Ecclesiastical Treasure

Treasure UUID	4e0d69d1189f4500a8548fe3ffa59c6e
Title (English)	St. George
Appellation (English)	Fresco by the church "St. George" in Kremikovski monastery Bulgaria
Media File UUID	c4622db1-321e-4655-a3de-8fe6bf6d0200
Media Type	photo
Media File	
New Media File *	Drop files here or click to upload Upload only 1 file

Figure 19: Update media page allows end-users to update existing media files of a particular ecclesiastical treasure.



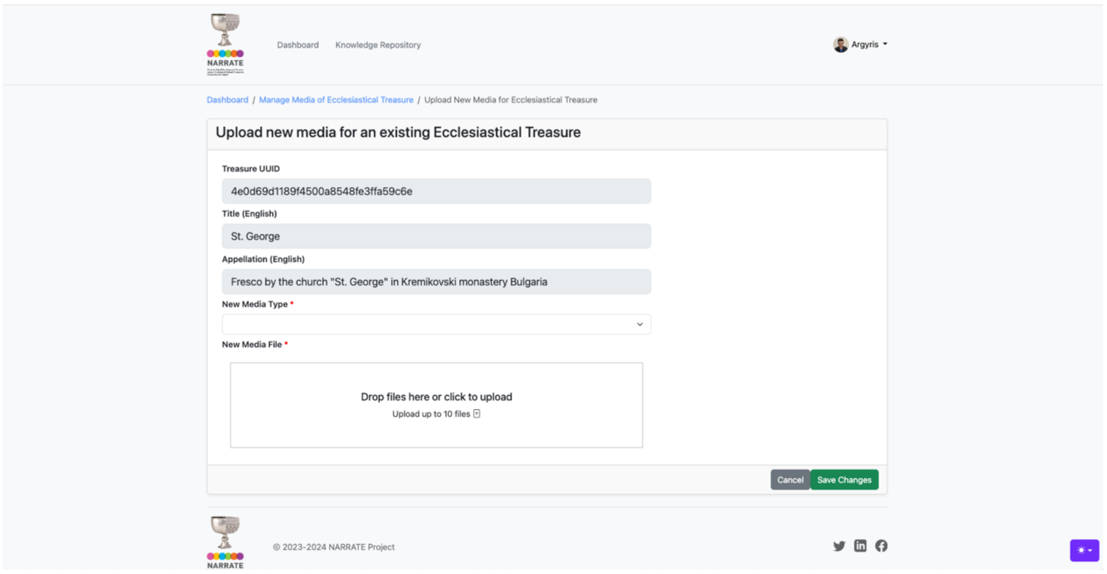
Dashboard / Manage Media of Ecclesiastical Treasure / Delete Media of Ecclesiastical Treasure

Delete existing media of Ecclesiastical Treasure

Treasure UUID	4e0d69d1189f4500a8548fe3ffa59c6e
Title (English)	St. George
Appellation (English)	Fresco by the church "St. George" in Kremikovski monastery Bulgaria
Media File UUID	c4622db1-321e-4655-a3de-8fe6bf6d0200
Media Type	photo
Media File	

Figure 20: Delete media page allows end-users to review and confirm the deletion of the media file of a particular ecclesiastical treasure.

The media management page also contains the “Upload New Media for Ecclesiastical Treasure” button, which redirects end-users to another page that allows them to upload new media files for an existing ecclesiastical treasure (**Figure 21**). The individual is requested to fill in the required fields (*i.e.*, new media type, new media file) in the upload new media form. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen. Upon successful completion of the action, the newly uploaded media file gets stored and becomes associated with the respective ecclesiastical treasure.


Dashboard
Knowledge Repository
Argyris
Dashboard / Manage Media of Ecclesiastical Treasure / Upload New Media for Ecclesiastical Treasure

Upload new media for an existing Ecclesiastical Treasure

Treasure UUID
4e0d69d1189f4500a8548fe3ffa59c6e

Title (English)
St. George

Appellation (English)
Fresco by the church "St. George" in Kremikovski monastery Bulgaria

New Media Type *

New Media File *

Drop files here or click to upload
Upload up to 10 files

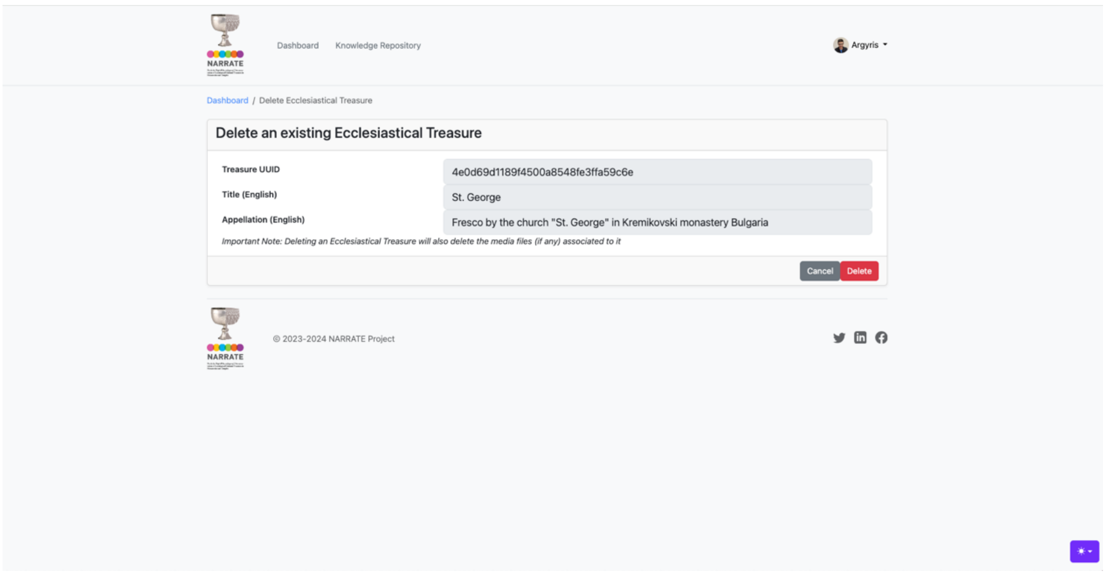
Cancel
Save Changes
© 2023-2024 NARRATE Project

Figure 21: Upload new media page allows end-users to upload new media files for an existing ecclesiastical treasure.

4.2.4 Delete existing ecclesiastical treasure (requires Sign in)

The delete existing ecclesiastical treasure page (**Figure 22**) of the NARRATE Interactive Dashboard allows individuals to review and confirm the deletion of the ecclesiastical treasure and its associated media. In case of unsuccessful action, the end-user is informed through an error message displayed on the screen. Upon successful completion of the action, the ecclesiastical treasure and its associated media get permanently deleted.

Co-funded by the European Union


Dashboard
Knowledge Repository
Argyris
Dashboard / Delete Ecclesiastical Treasure

Delete an existing Ecclesiastical Treasure

Treasure UUID	4e0d69d1189f4500a8548fe3ffa59c6e
Title (English)	St. George
Appellation (English)	Fresco by the church "St. George" in Kremikovski monastery Bulgaria

Important Note: Deleting an Ecclesiastical Treasure will also delete the media files (if any) associated to it

Cancel
Delete
© 2023-2024 NARRATE Project

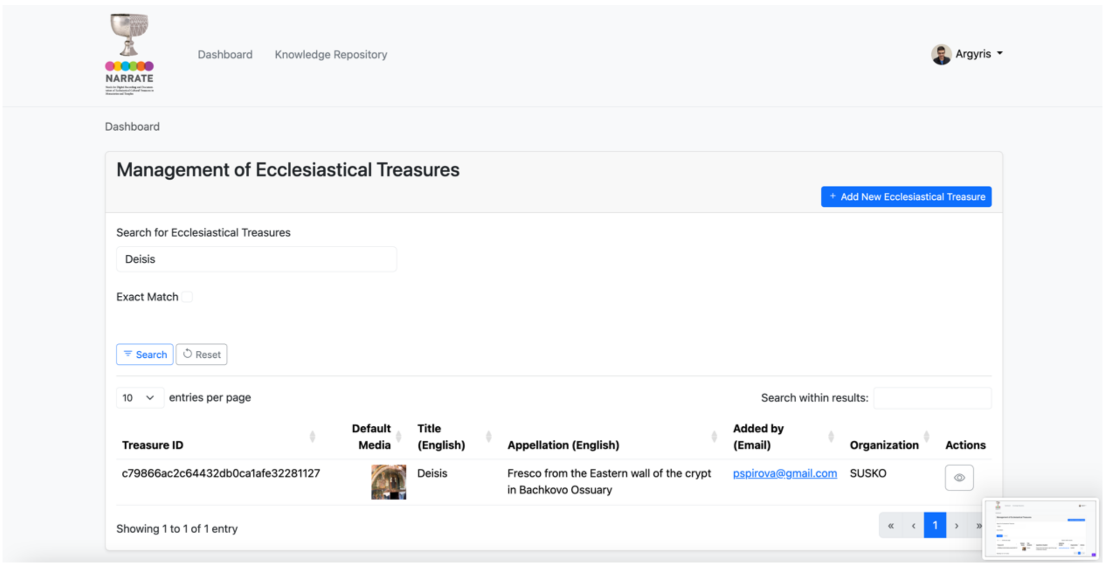
Figure 22: Delete existing ecclesiastical treasure page allows individuals to review and confirm the deletion of the ecclesiastical treasure and its associated media.

4.2.5 Advanced search for ecclesiastical treasures (requires Sign in)

The functionality of advanced search for ecclesiastical treasures (**Figure 23**) of the NARRATE Interactive Dashboard allows individuals to perform search operations by entering free text queries, which will search for a potential search result across all the descriptive details of the ecclesiastical treasures. The end-users also have the ability to tick the “Exact Match” checkbox, which will apply an exact matching for the keywords used in the search area.

Co-funded by the European Union

Contract number: 2022-1-EL01-KA220-HED-000089867



Management of Ecclesiastical Treasures

+ Add New Ecclesiastical Treasure

Search for Ecclesiastical Treasures: Deisis

Exact Match

Search | Reset

10 entries per page — Search within results:

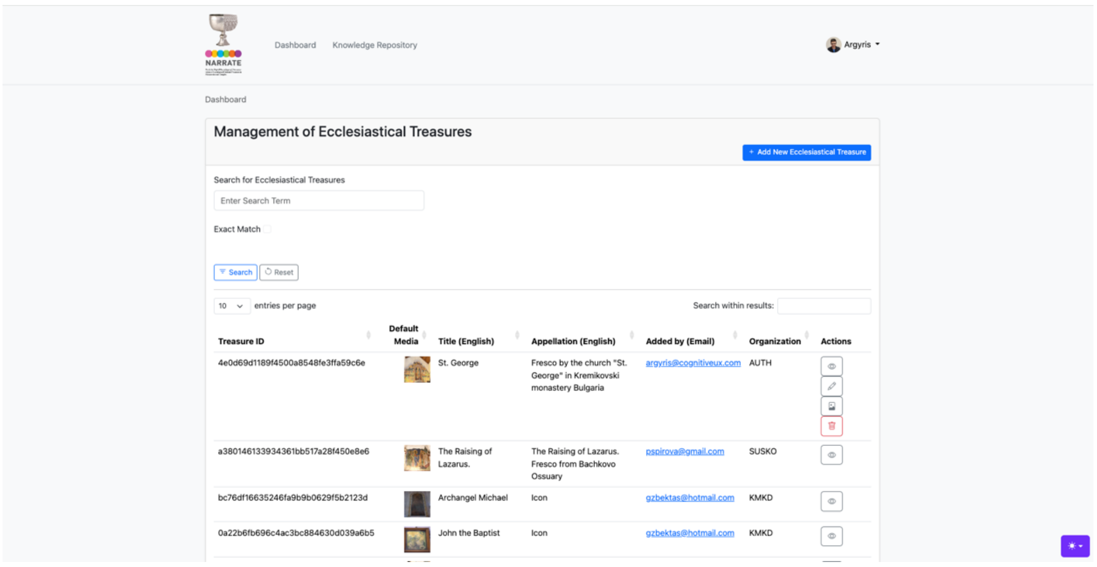
Treasure ID	Default Media	Title (English)	Appellation (English)	Added by (Email)	Organization	Actions
c79866ac2c64432db0ca1afe32281127		Deisis	Fresco from the Eastern wall of the crypt in Bachkovo Ossuary	pspirova@gmail.com	SUSKO	

Showing 1 to 1 of 1 entry

Figure 23: Advanced search for ecclesiastical treasures allows individuals to perform search operations by entering free text queries or performing exact match for the keywords used in the search area.

4.2.6 Basic access control for ecclesiastical treasures (requires Sign in)

The functionality of basic access control for the management of ecclesiastical treasures (**Figure 24**) of the NARRATE Interactive Dashboard handles the actions allowed to be performed by each signed in end-user. The access control mechanism allows end-users to see all the ecclesiastical treasures (including their descriptive details and their associated media) but they are allowed to update/delete only the ecclesiastical treasures (including their descriptive details and their associated media) that were added by them.



Dashboard

Management of Ecclesiastical Treasures

Search for Ecclesiastical Treasures

Exact Match

10 entries per page

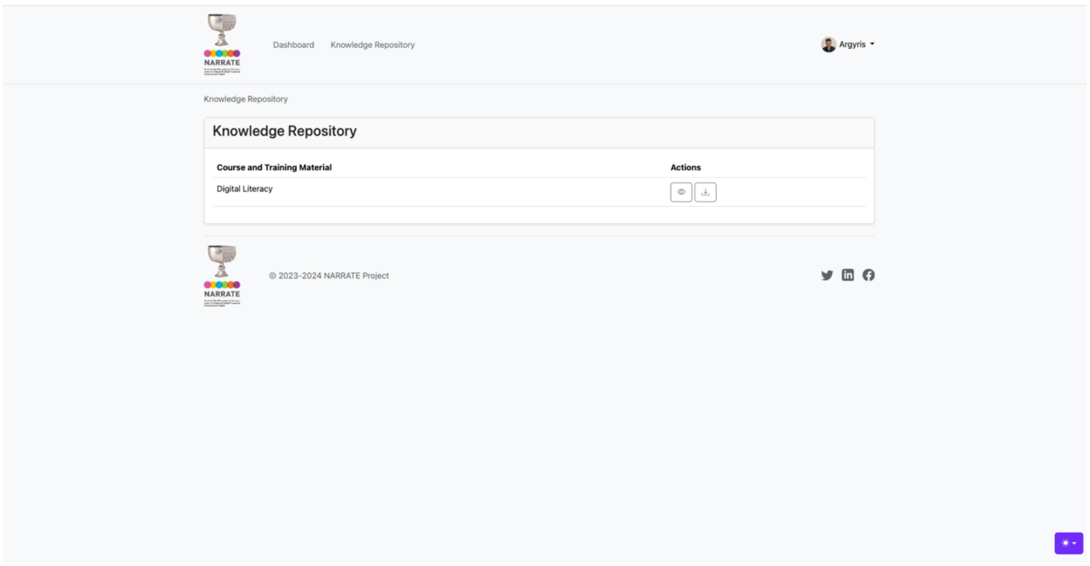
Search within results:

Treasure ID	Default Media	Title (English)	Appellation (English)	Added by (Email)	Organization	Actions
4e0d69d1189f4500a8548fe3ffa59c6e		St. George	Fresco by the church "St. George" in Kremikovski monastery Bulgaria	argyris@cognitiveux.com	AUTH	
a380146133934361bb517a28f450e8e6		The Raising of Lazarus.	The Raising of Lazarus. Fresco from Bachkovo Ossuary	pspirova@gmail.com	SUSKO	
bc76df16635246fa9b9b0629f5b2123d		Archangel Michael	Icon	gzbektas@hotmail.com	KMKD	
0a22b6fb696c4ac3bc884630d039a6b5		John the Baptist	Icon	gzbektas@hotmail.com	KMKD	

Figure 24: Basic access control for the management of ecclesiastical treasures, allowing end-users to see all the ecclesiastical treasures but update/delete only the eclesiastical treasures that were added by them.

4.3 Knowledge Repository (requires Sign in)

The knowledge repository page (**Figure 25**) of the NARRATE Interactive Dashboard serves as the hub for the project's educational and dissemination material regarding the digital recording and documentation of ecclesiastical treasures. Example resources available through this componente will include: *i)* Course material, which will offer curated content to support knowledge acquisition; *ii)* Training material, which will provide a systematic approach to learning and will aim to ensure competence; *iii)* Recorded webinars, which will offer in-depth insights and expertise; *iv)* Demonstration videos, which will provide practical demonstrations and will aim to enhance understanding and acceptance of the digital tools; and *v)* Dissemination material, which will act as the channel for sharing insights and expertise, promoting collaborations, and fostering a culture of continuous learning and knowledge exchange. End-users can either view or download the material provided in the knowledge repository page.



Knowledge Repository

Knowledge Repository

Course and Training Material	Actions
Digital Literacy	

© 2023-2024 NARRATE Project

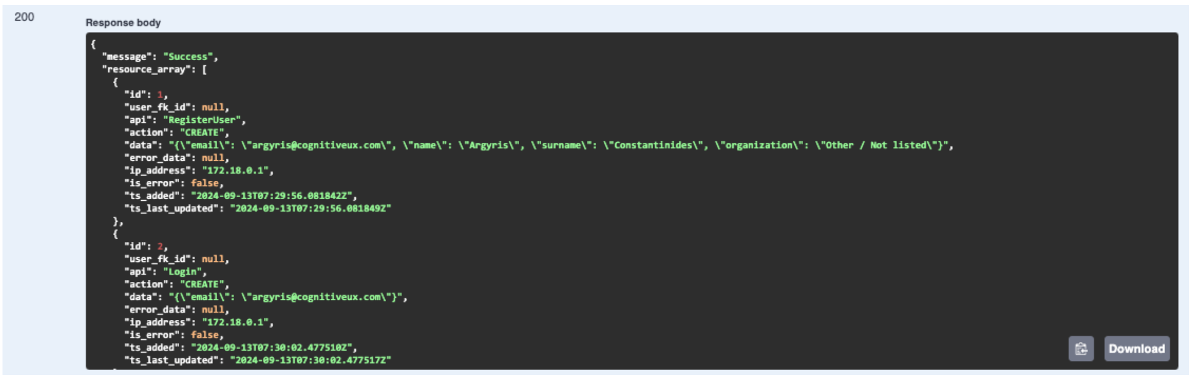
Figure 25: Knowledge repository page serves as the hub for the project's educational and dissemination material regarding the digital recording and documentation of ecclesiastical treasures.

4.4 Analytics (requires Administrator account & Sign in)

The analytics functionality is responsible for logging certain actions that occur within the system. Example actions include the management of: *i)* end-users accounts (*e.g.*, account creation, sign in to the system, reset/update of password, update of profile details, sign out); *ii)* ecclesiastical treasures (*e.g.*, viewing, adding, updating, deleting, and searching for ecclesiastical treasures); and *iii)* errors that occurred during end-users' interactions. The analytics functionality was implemented as an API (*i.e.*, /system-logs/list/), and is accessible only to signed in administrator accounts. For this purpose, an additional script was implemented, which facilitates the setup and creation of administrator accounts. The analytics API is accessible via the interactive demo page (see *APPENDIX B – Application Programming Interfaces*) and returns all the logged entries. **Figure 26** illustrates an example of the logged actions returned to the system administrator by the analytics API.

Co-funded by the European Union

Contract number: 2022-1-EL01-KA220-HED-000089867


200

Response body

```
{
  "message": "Success",
  "resource_array": [
    {
      "id": 1,
      "user_fk_id": null,
      "api": "RegisterUser",
      "action": "CREATE",
      "data": "{\"email\": \"argyris@cognitiveux.com\", \"name\": \"Argyris\", \"surname\": \"Constantinides\", \"organization\": \"Other / Not listed\"}",
      "error_data": null,
      "ip_address": "172.18.0.1",
      "is_error": false,
      "ts_added": "2024-09-13T07:29:56.081842Z",
      "ts_last_updated": "2024-09-13T07:29:56.081849Z"
    },
    {
      "id": 2,
      "user_fk_id": null,
      "api": "Login",
      "action": "CREATE",
      "data": "{\"email\": \"argyris@cognitiveux.com\"}",
      "error_data": null,
      "ip_address": "172.18.0.1",
      "is_error": false,
      "ts_added": "2024-09-13T07:30:02.477510Z",
      "ts_last_updated": "2024-09-13T07:30:02.477517Z"
```

Figure 26. Example of the logged actions returned to the system administrator by the analytics functionality.

Contract number: 2022-1-EL01-KA220-HED-000089867

5. Conclusions

The aim of this software report is to present the activities on the development and integration of the NARRATE ecclesiastical-centered data repository and discovery service. The report begins with an overview of the NARRATE framework solution, followed by a discussion of its architectural design and the technology stack used. Then, it provides a detailed description of the implemented functionalities, along with the creation of the interactive dashboard, which was developed based on earlier design mockups. Finally, the appendices provide information regarding accessing the online NARRATE framework and offer a detailed description of the application programming interfaces. The findings of this report will serve as the basis for the summative evaluation user study of the NARRATE framework.

References

[1] NARRATE Project (2023). Project Result WP2 R5 – Production of a framework for best practices guides.

[2] NARRATE Project (2023). Project Result WP2 R7 – Production of a conceptual framework that will identify the key characteristics, functionalities affordances, and modalities of the digital tools NARRATE will develop.

Co-funded by the European Union

APPENDIX A – Access URLs to the NARRATE Framework

NARRATE Interactive Dashboard (Web Application): https://narrate-api.cognitiveux.net/backend/login/

NARRATE Project Open-source Software Code Repository: https://github.com/cognitiveux/narrate-service

APPENDIX B – Application Programming Interfaces

Interactive demo page: https://narrate-api.cognitiveux.net/backend/demo/

Documentation page: https://narrate-api.cognitiveux.net/backend/doc/

The endpoints for interacting with the NARRATE service
Contact Info: admin@cognitiveux.com
Version: v2
BasePath:/backend
BSD License
http://apache.org/licenses/LICENSE-2.0.html

Access

1. APIKey KeyParamName:Authorization KeyInQuery:false KeyInHeader:true

Methods

[Jump to Models]

Table of Contents

- `POST /ecclesiastical-treasures/create/`
- `DELETE /ecclesiastical-treasures/delete/`
- `GET /ecclesiastical-treasures/fetch/`
- `GET /ecclesiastical-treasures/list/`
- `DELETE /ecclesiastical-treasures/media/delete/`
- `GET /ecclesiastical-treasures/media/list/`
- `POST /ecclesiastical-treasures/media/update/`
- `POST /ecclesiastical-treasures/media/upload new/`
- `POST /ecclesiastical-treasures/update/`

FileManagement

- `POST /file-management/media/temp/add/`
- `DELETE /file-management/media/temp/delete/`

SystemLogs

- `GET /system-logs/list/`

AccountManagement

Up

`POST /account-management/login/`

(accountManagementLoginCreate)

Creates a JSON Web Token for login purpose if the provided credentials are correct

Consumes

This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

data Login (required)

Body Parameter —

Return type

Response body for status code 201

Example data

Content-Type: application/json

```
{"empty": false}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
201
JSON Web Token has been created successfully. The value is returned in `resource_obj`. Response body for status code 201
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
403
Forbidden. User is not activated. You must activate it to proceed. Response body for status code 403
404
User not found Response body for status code 403
405
Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401
500
Internal server error Response body for status code 401

Up
`GET /account-management/poll_reset_email_status/`

(**accountManagementPollResetEmailStatusList**)
Polls the status of the reset code `email`

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Query parameters
email (required)
Query Parameter — Email
Return type
Response body for status code 200
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

Co-funded by the European Union

- `application/json`

Responses
200
Success. The status is returned in `task_status`. Response body for status code 200
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
404
Celery task ID for reset code not found or User not found Response body for status code 404
405
Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401
500
Internal server error Response body for status code 401

Up

`POST /account-management/refresh_token/`

(**accountManagementRefreshTokenCreate**)
Uses the longer-lived refresh token to obtain another access token
Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body
data RefreshToken (required)
Body Parameter —
Return type
Response body for status code 201_1
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

201
JSON Web Token has been created successfully. The value is returned in `resource_str`. Response body for status code 201_1
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400_1
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
405
Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401
500
Internal server error Response body for status code 401

Up

`POST /account-management/register_user/`

(**accountManagementRegisterUserCreate**)
Creates a new NARRATE user instance
Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body
data RegisterUser (required)
Body Parameter —
Return type
Response body for status code 403
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 403
400

Co-funded by the European Union
Contract number: 2022-1-EL01-KA220-HED-000089867

Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
405
Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401
500
Internal server error Response body for status code 401

Up
`POST /account-management/request_password_reset_code/`
(**accountManagementRequestPasswordResetCodeCreate**)
Request a reset code for reset password of account via email
Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body
data RequestPasswordResetCode (required)
Body Parameter —
Return type
Response body for status code 403
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 403
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
404
User not found Response body for status code 403
405
Method not allowed Response body for status code 401
415

Unsupported media type Response body for status code 401
422
Request limit exceeded. Try again in <Integer> minutes. Response body for status code 403
500
Internal server error Response body for status code 401

Up
`POST /account-management/reset_password/`

(**accountManagementResetPasswordCreate**)
Resets the password if the provided password reset code matches the latest password reset code received via email

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body
data ResetPassword (required)
Body Parameter —
Return type
Response body for status code 403
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 403
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
404
User not found Response body for status code 403
405
Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401

422
Reset code has expired or Reset code is incorrect or Reset code not requested Response body for status code 422
500
Internal server error Response body for status code 401

Up
POST /account-management/update_password/

(accountManagementUpdatePasswordCreate)
Updates the user's password

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body
data UpdatePassword (required)
Body Parameter —
Return type
Response body for status code 403
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 403
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
404
User not found Response body for status code 403
405
Method not allowed Response body for status code 401
415

Unsupported media type Response body for status code 401
422
Password is incorrect Response body for status code 403
500
Internal server error Response body for status code 401

Up
`POST /account-management/update_profile/`

(accountManagementUpdateProfileCreate)
Updates the user's profile details
Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body
data UpdateProfile (required)
Body Parameter —
Return type
Response body for status code 403
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 403
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
405
Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401
500

Co-funded by the European Union

Contract number: 2022-1-EL01-KA220-HED-000089867

Internal server error Response body for status code 401

EcclesiasticalTreasures

Up
`POST /ecclesiastical-treasures/create/`

(**ecclesiasticalTreasuresCreateCreate**)
Creates a new ecclesiastical treasure

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body
data EcclesiasticalTreasuresCreate (required)
Body Parameter —

Return type
Response body for status code 403

Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 403
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
405
Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401
500
Internal server error Response body for status code 401

Up

`DELETE /ecclesiastical-treasures/delete/`

(**ecclesiasticalTreasuresDeleteDelete**)

Delete data of an ecclesiastical treasure based on the `treasure_id`

Consumes

This API call consumes the following media types via the Content-Type request header:

- `application/json`

Query parameters

treasure_id (required)

Query Parameter — The uuid of the ecclesiastical treasure you would like to delete

Return type

Response body for status code 403

Example data

Content-Type: application/json

```
{"empty": false}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200

Success Response body for status code 403

400

Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400

401

Unauthorized - The request lacks valid authentication credentials. Response body for status code 401

403

Forbidden. You are not allowed to access this resource. Response body for status code 401

404

Ecclesiastical Treasure not found Response body for status code 403

405

Method not allowed Response body for status code 401

415

Unsupported media type Response body for status code 401

500
Internal server error Response body for status code 401

Up
GET /ecclesiastical-treasures/fetch/

(ecclesiasticalTreasuresFetchList)
Returns the data of a specific ecclesiastical treasure based on the `treasure_id`

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Query parameters
treasure_id (required)
Query Parameter — The uuid of the ecclesiastical treasure you would like to fetch

Return type
Response body for status code 200_1

Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 200_1
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
404
Ecclesiastical Treasure not found Response body for status code 403
405
Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401
500

Internal server error Response body for status code 401

Up
`GET /ecclesiastical-treasures/list/`

(ecclesiasticalTreasuresListList)
Returns the list of all ecclesiastical treasures based on the `search_keyword` and `exact_match` if given

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Query parameters
search_keyword (optional)
Query Parameter — The search keyword to filter ecclesiastical treasures
exact_match (optional)
Query Parameter — Whether the search keyword to be exact match or not when filtering ecclesiastical treasures

Return type
Response body for status code 200_2

Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 200_2
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
405
Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401
500

Internal server error [Response body for status code 401]

Up

DELETE /ecclesiastical-treasures/media/delete/

(ecclesiasticalTreasuresMediaDeleteDelete)

Delete media file of an ecclesiastical treasure based on the `uuid`

Consumes

This API call consumes the following media types via the Content-Type request header:

- `application/json`

Query parameters

treasure_id (required)

Query Parameter — The uuid of the ecclesiastical treasure

media_id (required)

Query Parameter — The uuid of the media file you would like to delete

Return type

[Response body for status code 200_3]

Example data

Content-Type: application/json

```
{"empty": false}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200

Success [Response body for status code 200_3]

400

Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned [Response body for status code 400]

401

Unauthorized - The request lacks valid authentication credentials. [Response body for status code 401]

403

Forbidden. You are not allowed to access this resource. [Response body for status code 401]

404

Ecclesiastical Treasure not found or Media File not found [Response body for status code 200_3]

405

Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401
500
Internal server error Response body for status code 401

Up
`GET /ecclesiastical-treasures/media/list/`

(**ecclesiasticalTreasuresMediaListList**)
Returns the list of all media for the given ecclesiastical treasure based on the `treasure_id`

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Query parameters
treasure_id (required)
Query Parameter — The uuid of the ecclesiastical treasure for which you would like to get its media files
Return type
Response body for status code 200_2
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 200_2
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
404
Ecclesiastical Treasure not found Response body for status code 403
405
Method not allowed Response body for status code 401

415
Unsupported media type Response body for status code 401
500
Internal server error Response body for status code 401

Up
`POST /ecclesiastical-treasures/media/update/`

(**ecclesiasticalTreasuresMediaUpdateCreate**)
Updates the media file of an ecclesiastical treasure
Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body
data EcclesiasticalTreasuresMediaUpdate (required)
Body Parameter —
Return type
Response body for status code 200_3
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 200_3
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
403
Forbidden. You are not allowed to access this resource. Response body for status code 401
404
Ecclesiastical Treasure not found or Media File not found Response body for status code 200_3

405
Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401
500
Internal server error Response body for status code 401

Up
`POST /ecclesiastical-treasures/media/upload_new/`

(**ecclesiasticalTreasuresMediaUploadNewCreate**)
Uploads new media for an ecclesiastical treasure

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body
data EcclesiasticalTreasuresMediaUploadNew **(required)**
Body Parameter —
Return type
Response body for status code 200_3
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 200_3
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
403
Forbidden. You are not allowed to access this resource. Response body for status code 401
404

Ecclesiastical Treasure not found or Media File not found Response body for status code 200_3
405
Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401
500
Internal server error Response body for status code 401

Up

`POST /ecclesiastical-treasures/update/`

(**ecclesiasticalTreasuresUpdateCreate**)
Updates an ecclesiastical treasure

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body
data EcclesiasticalTreasuresUpdate **(required)**
Body Parameter —

Return type
Response body for status code 403

Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 403
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
403

Forbidden. You are not allowed to access this resource. Response body for status code 401
404
Ecclesiastical Treasure not found Response body for status code 403
405
Method not allowed Response body for status code 401
415
Unsupported media type Response body for status code 401
500
Internal server error Response body for status code 401

FileManagement

Up
`POST /file-management/media/temp/add/`

(fileManagementMediaTempAddCreate)
Creates a new temp media entry based on the file received
Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/x-www-form-urlencoded`
- `multipart/form-data`

Form parameters
file_src (optional)
Form Parameter —
uuid (optional)
Form Parameter —
file_ext (optional)
Form Parameter —
Return type
Response body for status code 200_1
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200
Success Response body for status code 200_1
400
Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
404
User not found
405
Method not allowed Response body for status code 401
500
Internal server error Response body for status code 401

Up

`DELETE /file-management/media/temp/delete/`

(fileManagementMediaTempDeleteDelete)
Deletes data of a temp media file based on the `file_id`

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Query parameters
file_id (required)
Query Parameter — The uuid of the temp media file you would like to delete
Return type
Response body for status code 403
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 403
400

Bad request (Invalid data) - Any missing, already existing or bad formatted fields will be returned Response body for status code 400
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
404
Media File not found Response body for status code 403
405
Method not allowed Response body for status code 401
500
Internal server error Response body for status code 401

SystemLogs

Up
`GET /system-logs/list/`
(systemLogsListList)
Returns the list of all system logs
Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Return type
Response body for status code 200_2
Example data
Content-Type: application/json

```
{"empty": false}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Success Response body for status code 200_2
401
Unauthorized - The request lacks valid authentication credentials. Response body for status code 401
403

Co-funded by the European Union

Forbidden. You are not allowed to access this resource. Response body for status code 401

405

Method not allowed Response body for status code 401

415

Unsupported media type Response body for status code 401

500

Internal server error Response body for status code 401

Models

[Jump to Methods]

Table of Contents

`Details for updating media of ecclesiastical treasures` -Up

treasure_id (optional)
String
old_media_id (optional)
String
new_media_id (optional)
String

`Details for updating the user profile` -Up

name (optional)
String
surname (optional)
String
telephone (optional)
String
media_type_id (optional)
String
type (optional)
String

`Details for uploading new media for the ecclesiastical treasures` -Up

treasure_id (optional)
String
media_type_id (optional)
String
type (optional)
String

`EcclesiasticalTreasuresCreate` -Up

title_en
String
title_gr (optional)
String
title_bg (optional)
String
title_tk (optional)
String
appellation_en
String
appellation_gr (optional)
String
appellation_bg (optional)

Co-funded by the European Union

Contract number: 2022-1-EL01-KA220-HED-000089867

String
appellation_tk (optional)
String
existing_obj_code (optional)
String
desc_short_version (optional)
String
desc_extended_version (optional)
String
time_span (optional)
String
kind (optional)
String
creator (optional)
String
beginning_of_existence (optional)
String
was_in_church (optional)
Boolean
was_in_another_country (optional)
Boolean
was_lost_and_found (optional)
Boolean
dimension (optional)
String
material (optional)
String
inscription (optional)
String
manuscript_text (optional)
String
event_information (optional)
String
previous_documentation (optional)
String
relevant_bibliography (optional)
String
preservation_status (optional)
String
conservation_status (optional)
String
group_of_objects (optional)
array[String]
collection_it_belongs (optional)
String
position_of_treasure (optional)
String

people_that_help_with_documentation (optional)
array[String]

`EcclesiasticalTreasuresMediaUpdate` - The details of the ecclesiastical treasure media you would like to updateUp

updating_data (optional)
Details for updating media of ecclesiastical treasures

`EcclesiasticalTreasuresMediaUploadNew` - The details of the ecclesiastical treasure media you would like to uploadUp

uploading_data (optional)
Details for uploading new media for the ecclesiastical treasures

`EcclesiasticalTreasuresUpdate` -Up

uuid
String
title_en
String
title_gr (optional)
String
title_bg (optional)
String
title_tk (optional)
String
appellation_en
String
appellation_gr (optional)
String
appellation_bg (optional)
String
appellation_tk (optional)
String
existing_obj_code (optional)
String
desc_short_version (optional)
String
desc_extended_version (optional)
String
time_span (optional)
String
kind (optional)
String
creator (optional)

Co-funded by the European Union

String
beginning_of_existence (optional)
String
was_in_church (optional)
Boolean
was_in_another_country (optional)
Boolean
was_lost_and_found (optional)
Boolean
dimension (optional)
String
material (optional)
String
inscription (optional)
String
manuscript_text (optional)
String
event_information (optional)
String
previous_documentation (optional)
String
relevant_bibliography (optional)
String
preservation_status (optional)
String
conservation_status (optional)
String
group_of_objects (optional)
array[String]
collection_it_belongs (optional)
String
position_of_treasure (optional)
String
people_that_help_with_documentation (optional)
array[String]

`Login` -Up

email
String
password
String

`RefreshToken` -Up

refresh

Co-funded by the European Union

String
access (optional)
String

`RegisterUser` -Up

email
String format: email
name
String
organization
String
Enum:
AUTH
IHU
KMKD
SUSKO
Other / Not listed
password
String
surname
String

`RequestPasswordResetCode` -Up

email
String format: email

`ResetPassword` -Up

email
String format: email
password
String
reset_code
String

`Response body for status code 200` -Up

Following keys are returned as json
message (optional)
String A general message description
task_status (optional)
String The status of the task: ['PENDING', 'SUCCESS', 'FAILURE']

`Response body for status code 200_1` -Up

Following keys are returned as json
message (optional)
String A general message description
resource_obj (optional)
Object A dictionary that contains the JWT in the form of key-value pairs. The key `access` is a string that corresponds to the JWT access token and the key `refresh` is a string that corresponds to the JWT refresh token.

`Response body for status code 200_2` -Up

Following keys are returned as json
message (optional)
String A general message description
resource_array (optional)
array[String] An array with all the available data

`Response body for status code 200_3` -Up

Following keys are returned as json
message (optional)
String A general message description
resource_name (optional)
String The name of the resource
Enum:
ecclesiastical_treasure
media_file

`Response body for status code 201` -Up

Following keys are returned as json
message (optional)
String A general message description
resource_name (optional)
String The name of the resource
resource_obj (optional)
Object A dictionary that contains the JWT in the form of key-value pairs. The key `access` is a string that corresponds to the JWT access token and the key `refresh` is a string that corresponds to the JWT refresh token.

`Response body for status code 201_1` -Up

Following keys are returned as json
message (optional)
String A general message description
resource_name (optional)
String The name of the resource

Co-funded by the European Union

resource_str (optional)
String A string value associated with the resource_name

`Response body for status code 400` -Up

Following keys are returned as json
message (optional)
String A general message description
bad_formatted_fields (optional)
array[String] Any field that is not in the correct format will be returned in the list
missing_required_fields (optional)
array[String] The missing required fields are returned as a list
already_exists_fields (optional)
array[String] Any field that is unique and already exists, will be returned in the list
error_details (optional)
Object A dictionary that contains descriptive information about the validation errors in the form of key-value pairs. Each key is a string that corresponds to the problematic field and the associated value is a list of strings that contains the error details. If a JSON parse error occurred, there will be only one key named `json`.

`Response body for status code 400_1` -Up

Following keys are returned as json
message (optional)
String A general message description
bad_formatted_fields (optional)
array[String] Any field that is not in the correct format will be returned in the list
missing_required_fields (optional)
array[String] The missing required fields are returned as a list
error_details (optional)
Object A dictionary that contains descriptive information about the validation errors in the form of key-value pairs. Each key is a string that corresponds to the problematic field and the associated value is a list of strings that contains the error details. If a JSON parse error occurred, there will be only one key named `json`.

`Response body for status code 401` -Up

Following keys are returned as json
message (optional)
String A general message description

`Response body for status code 403` -Up

Following keys are returned as json
message (optional)
String A general message description
resource_name (optional)
String The name of the resource

`Response body for status code 404` **-Up**

Following keys are returned as json
message (optional)
String A general message description
resource_name (optional)
String The name of the resource
Enum:
c_reset_task_id
user

`Response body for status code 422` **-Up**

Following keys are returned as json
message (optional)
String A general message description
resource_name (optional)
String The name of the resource
reason (optional)
String The reason behind this error message
Enum:
expired_reset_code
incorrect_reset_code
not_requested_reset_code

`UpdatePassword` **-Up**

current_password
String
new_password
String

`UpdateProfile` **- The details of the user profile you would like to updateUp**

updating_data (optional)
Details for updating the user profile