This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n: 761999

EasyTV: Easing the access of Europeans with disabilities to converging media and content.

D5.2 – Mid-term report on the set up and implementation of the EasyTV crowdsourcing sign language platform and repository

EasyTV Project

H2020. ICT-19-2017 Media and content convergence. – IA Innovation action.

Grant Agreement n°: 761999

Start date of project: 1 Oct. 2017

Duration: 30 months

Document. ref.: Deliverable 5.2

Disclaimer

The EasyTV Consortium consists of the following partners:

	Partner Name	Short name	Country
1	Universidad Politécnica de Madrid	UPM	ES
2	Engineering Ingegneria Informatica S.P.A.	ENG	IT
3	Centre for Research and Technology Hellas/Information Technologies Institute	CERTH	GR
4	Mediavoice SRL	MV	IT
5	Universitat Autònoma Barcelona	UAB	ES
6	Corporació Catalana de Mitjans Audiovisuals SA	CCMA	ES
7	ARX.NET SA	ARX	GR
8	Fundación Confederación Nacional Sordos España para la supresión de barreras de comunicación	FCNSE	ES
9	Unione Italiana dei ciechi e degli ipovedenti	UICI	IT

PROGRAMME NAME:	H2020. ICT-19-2017 Media and content convergence - IA Innovation action
PROJECT NUMBER:	761999
PROJECT TITLE:	EASYTV
RESPONSIBLE UNIT:	CERTH
INVOLVED UNITS:	CERTH
DOCUMENT NUMBER:	D5.2
DOCUMENT TITLE:	Mid-term report on the set up and implementation of the EasyTV crowdsourcing sign language platform and repository
WORK-PACKAGE:	WP5
DELIVERABLE TYPE:	R
CONTRACTUAL DATE OF DELIVERY:	30-10-2018
LAST UPDATE:	20-10-2018
DISTRIBUTION LEVEL:	PU

Distribution level:

PU = *Public,*

RE = *Restricted to a group of the specified Consortium,*

PP = *Restricted to other program participants (including Commission Services),*

CO = *Confidential, only for members of the LASIE Consortium (including the Commission Services)*

Document History

VERSION	DATE	STATUS	AUTHORS, REVIEWER	DESCRIPTION
V0.1	12/09/2018	Draft	Thanasis Kalvourtzis, (CERTH)	ToC
V0.2	15/09/2018	Draft	Thanasis Kalvourtzis (CERTH), Kiriakos Stefanidis (CERTH), Kosmas Dimitropoulos (CERTH)	First Draft Version
V0.3	19/09/2018	Draft	Thanasis Kalvourtzis, (CERTH)	First Draft Version and Appendices
V0.4	22/09/2018	Final	Thanasis Kalvourtzis, (CERTH), Kosmas Dimitropoulos (CERTH)	Final version ready for internal review
V0.5	30/10/2018	Final	Thanasis Kalvourtzis, (CERTH), Kosmas Dimitropoulos (CERTH)	Final version ready for submission

Definitions, Acronyms and Abbreviations

ACRONYMS / ABBREVIATIONS	DESCRIPTION
API	Application programming interface
CSS	Cascading Style Sheet
FAQ	Frequently Asked Questions
HTML5	HyperText Markup Language
UI	User Interface
HTTP	HyperText Transfer Protocol
Mocap	Motion Capture
CLI	Command-line interface
SLT	Sign language task
JSON	JavaScript Object Notation

CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

This document is the preliminary version of deliverable D5.2.1 “Mid-term report on the set up and implementation of the EasyTV crowdsourcing Sign Language platform and repository”. The goal of this deliverable is to present the first outcomes of Task 5.2, i.e., the first version of the crowdsourcing sign language platform. The document describes the architecture, user roles and the entire work performed for the implementation of the EasyTV crowdsourcing platform during the first year of the project. More specifically, the main points outlined in this deliverable are the following:

- The architecture and the main components of the crowdsourcing platform along with the database structure.
- The different user roles supported by the crowdsourcing sign language platform.
- The main database entities supporting the creation, management and distribution of the crowdsourcing tasks.
- The front- and back-end systems of the platform.
- The APIs for the communication of the platform with the capturing module and the multilingual ontology.

Finally, details about the platform’s interaction with the rest of the EasyTV component-based system are presented in this document.

1. INTRODUCTION

In the context of component-based EasyTV platform, the crowdsourcing module is going to serve as a web platform, allowing users to participate in tasks and contribute with their knowledge-based skills to the creation of a multilingual sign language repository. Throughout this document, the methodologies, tools and workflows designed and developed for the crowdsourcing module during the first year of the EasyTV project are described. Moreover, the current state of the implementation of the crowdsourcing component is presented in detail. This deliverable comprises the basis for further development and refinement of the EasyTV crowdsourcing module that will be presented in deliverable D5.6 “Final report on the set up and implementation of the EasyTV crowdsourcing sign language platform and repository”.

The present document is organized as follows:

Chapter 2 provides an overview of the crowdsourcing concept, describes the way other systems took advantage of it and how the current application aims to leverage it.

Chapter 3 provides insight about the application’s architecture and describes clearly and in detail the software components, the user roles and requirements and the software stack.

Chapter 4 focuses on describing the front-end and back-end software stacks employed during the first stage of development of the EasyTV crowdsourcing module.

Chapter 5 analyzes the relationships between the EasyTV crowdsourcing platform and other modules developed in the framework of the EasyTV project.

Chapter 6 presents a graphical demonstration of the user interfaces and how these interfaces are related to the corresponding workflows.

Finally, Chapter 7 summarizes the work presented in this document and discusses about the forthcoming work steps for the further development and refinement of the EasyTV crowdsourcing platform.

2. CROWDSOURCING SYSTEMS OVERVIEW

In this section, we initially review existing crowdsourcing frameworks and present the definitions of crowdsourcing terminology used in the rest of the document. Subsequently, we present the general context of the Sign Language crowdsourcing platform and its interconnections with the other modules/components of the EasyTV system.

2.1. Definitions

To facilitate reading, in this sub-section, we provide the definitions of some of the most typical terms used in crowdsourcing systems.

***Task*:** A number of well-defined steps to be performed as constituents of a project.

Project: A collection of tasks.

***Moderator*:** The creator and manager of tasks.

***Worker*:** A simple user of the platform. It is assumed that a user contributes to projects by employing his/her knowledge and skills.

***Review*:** Content is open to review according to the policies of the corresponding project.

***Validation*:** Users with specific roles provide the validation of submitted data. These users are usually experts at manipulating and handling these particular types of data.

2.2. Examples of crowdsourcing applications

Crowdsourcing is an online, distributed problem-solving and production model that has emerged in recent years [1]. According to Howe [2], who first introduced this term, crowdsourcing is a sourcing model, in which organizations can develop applications that employ advanced internet technologies in order to harness the efforts and resources of a group of people, i.e. crowd[1] to perform specific tasks [3]. In contrast to traditional recruiting processes, where dedicated employees are selected and assigned to tasks by an employer, in crowdsourcing, the employer submits the task as an open call to a large anonymous crowd of workers. The workers can then freely decide which available task they want to work on [4].There are several advantages in using crowdsourcing models, such as improved costs, speed, quality, flexibility, scalability and diversity [5]. However, there are also significant challenges that arise by employing crowdsourcing systems. The most important of these challenges can be categorized as task related, i.e. design, routing, coordination and aggregation of tasks and crowd-related, i.e. motivation problems, incentive systems and quality control of work results [5]. Crowdsourcing is not bounded by specific constraints on the areas that it can be applied and, as a result, there are several successful implementations of crowdsourcing systems in several areas that are presented in the literature. A few important areas, in which crowdsourcing has been successfully applied are in journalism [6], policy making [7] and health [8].

Typical applications of the crowdsourcing concept are:

- Dataset collection and enhancement
- Dataset annotation
- Dataset categorization
- Sentiment analysis
- Other human-driven tasks

Apart from strictly task-based models, other web-based models of voluntary and collaborative knowledge synthesis and production has been established. Such models are proven successful based on simple yet fundamental innovations, i.e. open-source the publishing model and allow the

[1] https://searchcio.techtarget.com/definition/crowdsourcing

review work after publishing. Examples of such models are the wiki-like platforms that are based on knowledge creation to achieve a great deal in the empowering of the global Information Society. It has been observed that interested users are actively involved in the work of creating and synthesizing content, as well as the meta-work of reviewing, correcting, and organizing.

Finally, we should note the existence of modern web platforms that aim to crowd-source the experience of learning through interactive concepts. Probably the most successful example is that of Memrise [9], an online community that attempts to teach people foreign languages through user-generated content. The users participate in the platform by creating foreign-language lessons, enriching them with media content like memes and translated videos. The platform lets other users to customize the memes, in case they come up with alternative ideas. This open contribution model is able to lead to the creation of user-generated content of high quality. The high quality of the content is achieved by continuous user feedback that constantly improves the content.

2.3. Crowdsourcing application scope in the context of the EasyTV component-based system

The development of the EasyTV Crowdsourcing Platform is based upon a micro-tasking framework that has been specifically designed for providing infrastructure for crowdsourcing applications. It is a web platform that allows the launching and management of projects from the content owners through a web Graphical User Interface (GUI) or an Application Programming Interface (API). Additionally, volunteers-workers can contribute from their web browsers towards the completion of the projects. The EasyTV Crowdsourcing Platform is designed in a way that aims to provide an intuitive User Interface (UI) for the creation, distribution and assessment of crowdsourcing tasks.

More specifically the Crowdsourcing Platform aims to provide a number of functionalities to facilitate the sign language and subtitle production procedures with the collaboration of crowd workers. The platform includes functionalities for task definition, distribution and validation, and allows the creation and management of professional user profiles and the access of users in the sign language and subtitle production procedures through web GUIs. The interconnections of the Crowdsourcing Platform with the other EasyTV modules are depicted in Figure 1.

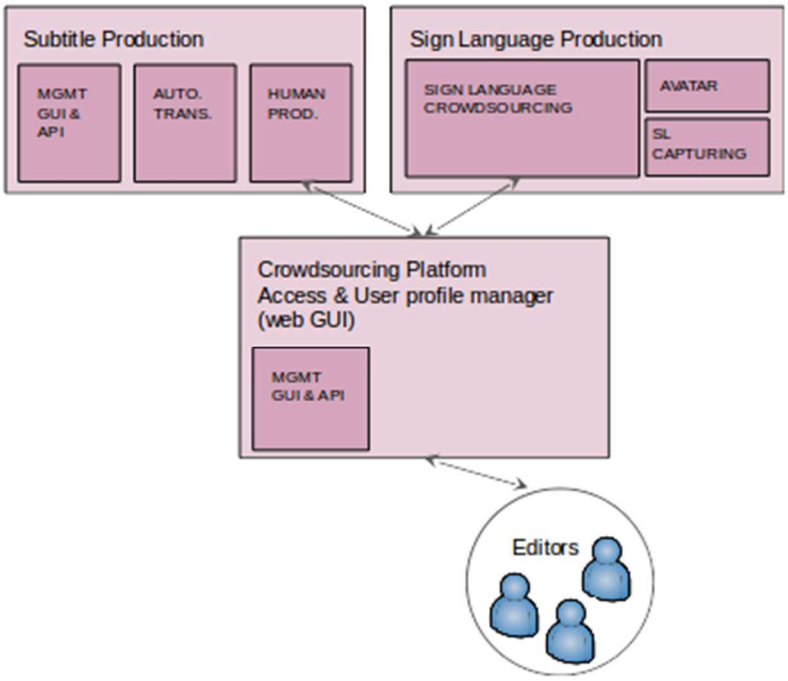


Figure 1: Crowdsourcing component positioning [10]

We should also note that one of the main requirements of the platform is the establishment of a two-way communication with the EasyTV ontology module *(as a service)*, which provides software infrastructure to process annotated sign language videos with Natural Language Processing (NLP) techniques. The ultimate goal is the creation of a multilingual knowledge base through the combination of annotated sign language videos that are created from users and stored in a repository with multilingual ontologies. Signs in different languages will be associated with their meaning in natural language in order to enrich the BabelNet – a multilingual encyclopaedic dictionary and semantic network[13].

3. PLATFORM ARCHITECTURE

This chapter focuses on the description of the architecture of the EasTV Crowdsourcing Platform. Initially, we define a high-level view of the main components of the platform. Afterwards, the user roles and scenarios in form of flowcharts are presented. These descriptions draw upon work from previous deliverables, in which the system requirements of the EasyTV platform, the user requirements and the EasyTV components and services were defined [10][11][12]. Moreover, in this chapter, we describe the main data models, which form the platform's main database entities.

3.1. Architecture components overview

The analysis of the user and system requirements leads to the creation of the first architecture of the EasyTV Crowdsourcing Platform that is outlined in Figure 2 with the help of a component diagram. The component diagram comprises a static implementation view of the system. It should be noted that these components and interfaces form a group of functionalities and services that is meant as a guideline to support the implementation perspective and does not represent a strict definition of isolated software modules.

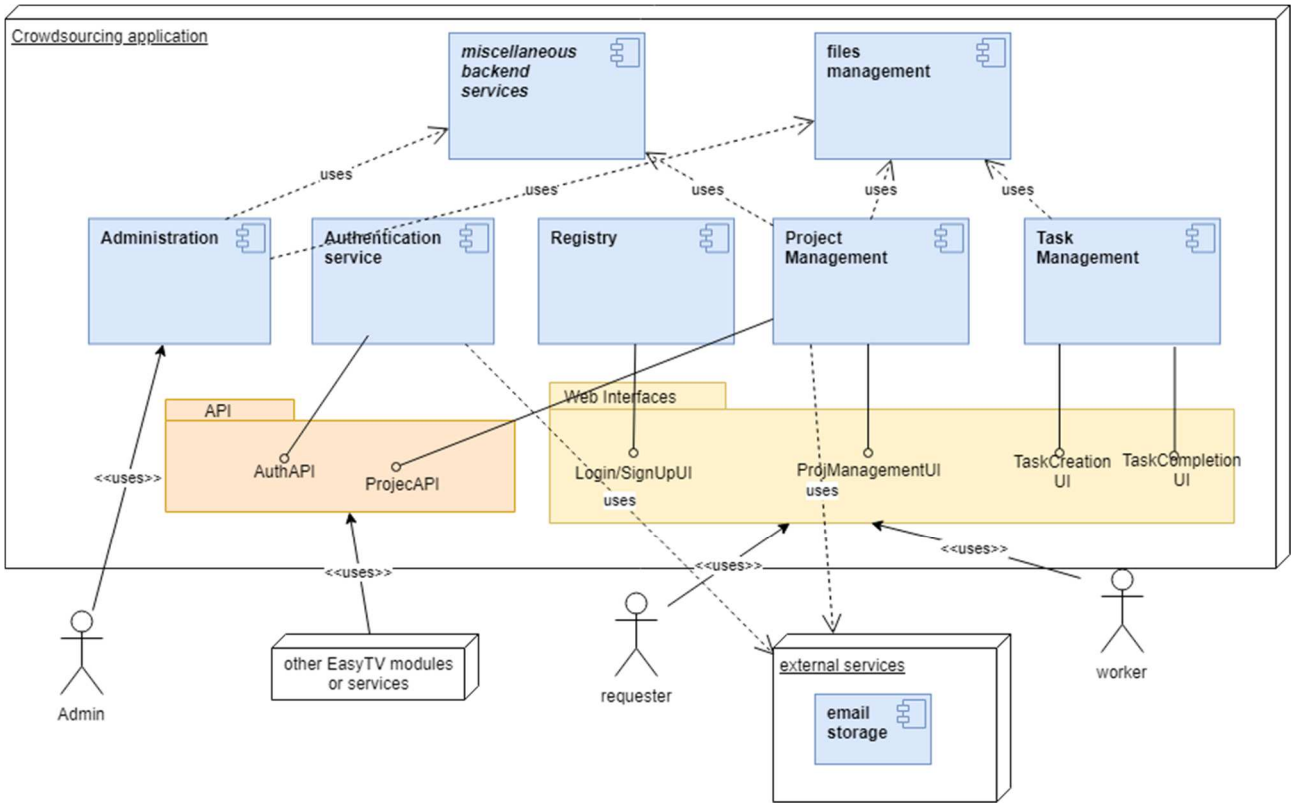


Figure 2: System Component Diagram

Administration component:

This component facilitates the administration functionalities, which are mainly concerned with the technical control and monitoring of the platform. The functionalities of this component are meant to be exposed via web UIs, like dashboard interfaces and settings to the administrators of the EasyTV platform.

Registry component:

The registry component is responsible for handling the registration of users or apps in the EasyTV Crowdsourcing platform. UIs are constructed and deployed for user registration and app registration through token generation and authentication.

Authentication component:

The authentication component has two main purposes: it handles user authentication and enforces the defined access policies for the different types of users. The implementation of this component incorporates well-known standards of user authentication via web token authentication.

Project management component

This component provides a number of functionalities related to project management. These functionalities are provided to end-users through user interfaces (or an API for external modules or services) in order to enable these users to define and manage projects.

Task creator/management component

This component provides a number of functionalities that are related to task creation and completion procedures. Examples of such procedures are task submission, validation of task completion results and storage of results in files. Different UIs are designed and developed for task editing and completion, while an API is provided for external apps.

Files management component

The files management component is responsible for handling operations that involve the creation of file streams for the storage and upload of content to the EasyTV Crowdsourcing Platform.

Miscellaneous services component

The miscellaneous services component includes a number of back-end helper services. An example of such services is the aggregation of statistics about task completion rates or user involvement in tasks. These services are not directly exposed through API endpoints.

API component

The API component is responsible for managing the endpoints for the communication of the EasyTV Crowdsourcing Platform with other modules.

External services

External services refer to a group of external components that offer services that are vital for the smooth operation of the platform, such as off-platform communication or data storage requirements.

3.2. Users and scenarios

In the context of a web application, such as a crowdsourcing and data-repository platform, the definition of users, their roles and their functionalities constitute a crucial part of the platform's development. Before any technical implementation is made, the users' roles and functionalities have been under heavy discussion from the EasyTV consortium. Furthermore, the deliverable D1.1 [11] gave us useful insights about the functionalities that users need in order to fulfill their role. Based on the discussions and the user requirements, the following user roles have been proposed and defined:

3.2.1. Administrator

The role of administrator is defined as the technical manager of the platform. In the first place, an administrator should be supported by an appropriate dashboard, which provides an overview of the current status of the platform regarding the statistics of platform's usage and resources. The administrator undertakes responsibilities related to technical aspects of the platform's operation. For

example, an administrator may configure the settings of the EasyTV Crowdsourcing Platform in order to enable the notification of users about upcoming upgrades of the platform.

3.2.2. **Workers**

Workers are the platform's contributing users. We currently describe two categories of users according to the kind of tasks that the platform will support based on its current scope.

3.2.2.1 Signer worker

These users constitute the main pool of contributors for the creation of a sign language database. Their participation in the platform is decided according to their language background that is defined by themselves during the registration procedure. The user should initially sign up to the platform by providing a minimum amount of information in order to be able to work in a project listed in the crowdsourcing platform. During the crowdsourced tasks, the platform prompts the user to follow a number of steps in order to fulfil a task by uploading content in the appropriate format. It also gives them rights to edit and review their own submissions or others' according to policies.

3.2.2.2 Subtitle worker

According to the specifications of the Subtitling Production Tool of the EasyTV platform, the crowdsourcing platform is intended to be a constituent part of the pipeline of the *Human subtitling production* module by fulfilling the role of a profile-management portal. For this purpose, the profiles of potential collaborative users are going to be kept and managed through an appropriate interface, which is meant to provide a broadcaster with access granting options to these users. The user/subtitle worker will come into contact with the content owner and will be tasked with performing translation requests by the content owner.

3.2.3. **Moderator of Sign Language Tasks**

The moderator of sign language tasks (SLTs) constitutes an advanced user of the EasyTV crowdsourcing platform. A moderator is most frequently the user that requests the creation and completion of crowdsourcing-based tasks. Through the platform's infrastructure, this user creates and manages Sign Language projects and is responsible for reviewing the progress of the tasks. The platform offers specific tools and content access for this purpose, like a dashboard.

3.2.4. **Moderator of content owner**

A moderator of content owner is a user of the EasyTV Crowdsourcing Platform that can be assigned by a broadcaster/content owner in order to be responsible for the assignment of reviewer and professional profiles to the corresponding users.

3.2.5. **User Scenarios**

In this section, we describe user scenarios that are designed and implemented for the EasyTV Crowdsourcing Platform in the framework of the EasyTV project. These scenarios are backed up by flowcharts that visualize the user roles and functionalities. In the following flowcharts, the involved shapes, along with the corresponding semantic information they carry, are presented in Appendix 1. It should be noted that these charts demonstrate a flow of asynchronous events as they involve human initiated actions.

It is a prerequisite that all categories of users have provided a minimum of personal information during the sign-up process. This information is necessary and is taken into account during processes like user-profiling or task-distribution.

Sign Language Task scenario:

A user that has a sign language background can log in the platform as a signer worker. User's UI displays a panel of active projects and their descriptions according to the logged user's background information. The user selects a project and the system responds with a view that lists the active tasks. The user proceeds to the task completion guided through steps by the GUI. At this stage, the system simply confirms that the submitted files have the correct format and meet the requirements of the task. After the successful completion of the task, the submissions of the user are stored in the system and labeled with status "under review". A moderator, who may be the creator of the project or assigned as reviewer of the project, gets notified about any progress related to tasks of his/her responsibility. In the event that a moderator validates the input of the signer worker for a task, the status of the submission of the task changes as accepted and is included in the repository. Finally, a user subscribed to a project should get automated notifications, when updates occur in the corresponding project.

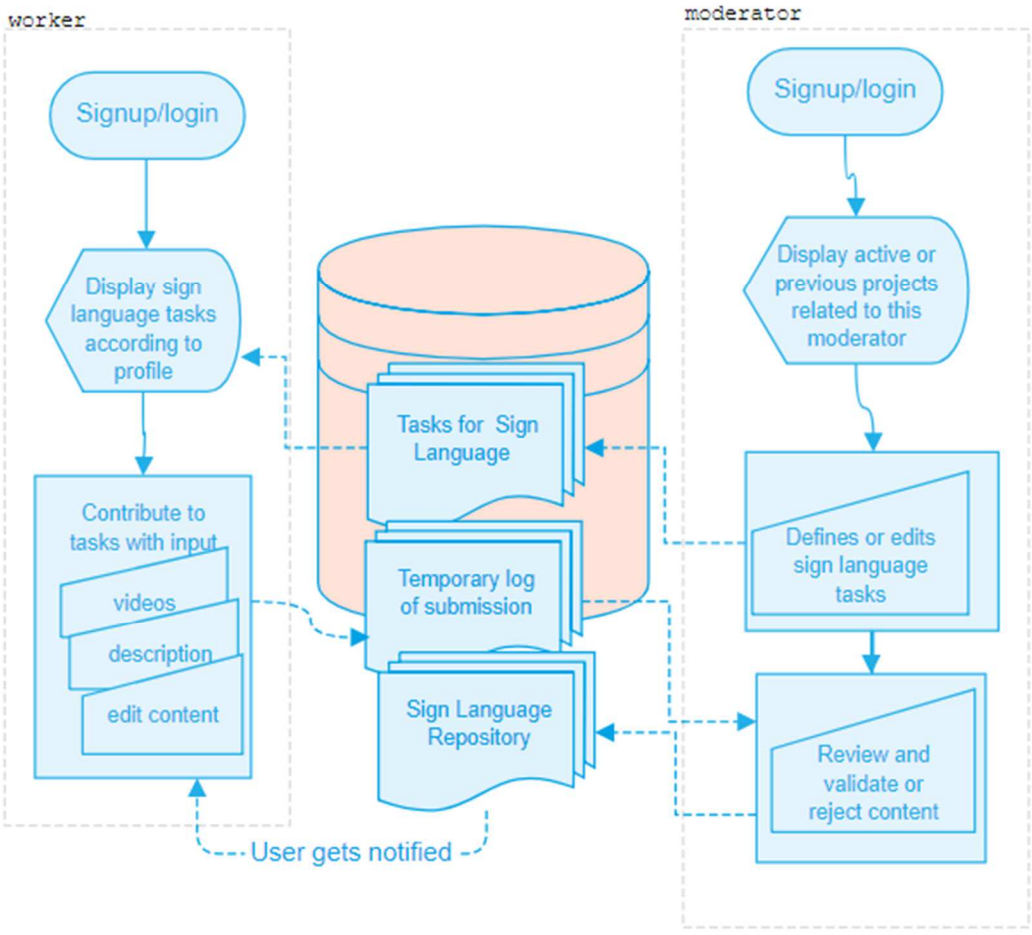


Figure 3: Scenario - Task workflow

Human subtitling module scenario:

Users are registered in the platform as potential collaborators and have also provided information about their language background and their competence. The broadcaster user manages different user's trust levels through a web GUI and also manages content access rights and quality assigning trust levels. Users proceed to contribute to a task and the platform assists them in accordance with the trust level of the author user of the translation. The broadcaster manages access to contents

with subtitles in its original language. Once the translated material is ready, the broadcaster is able to download it. Neither the original nor the translated content is to be made publicly available directly from the EasyTV Crowdsourcing Platform due to rights management. The publication of this content will be performed exclusively by the broadcaster over its own publication platform. Finally, the EasyTV Crowdsourcing Platform provides alert messages to the users, when additional content that requires translation is uploaded and to the broadcaster, when a translation task is finished.

Non-User Scenarios

The following description constitutes a non-user scenario, in the sense that it describes interaction between EasyTV software modules. The flow presented in this chart is related to the scope and functionality of the EasyTV Ontology module, as described in D3.2.1 [13]. In that deliverable, the objective of the enrichment of the multilingual ontology is described in relation to the EasyTV Sign Language repository.

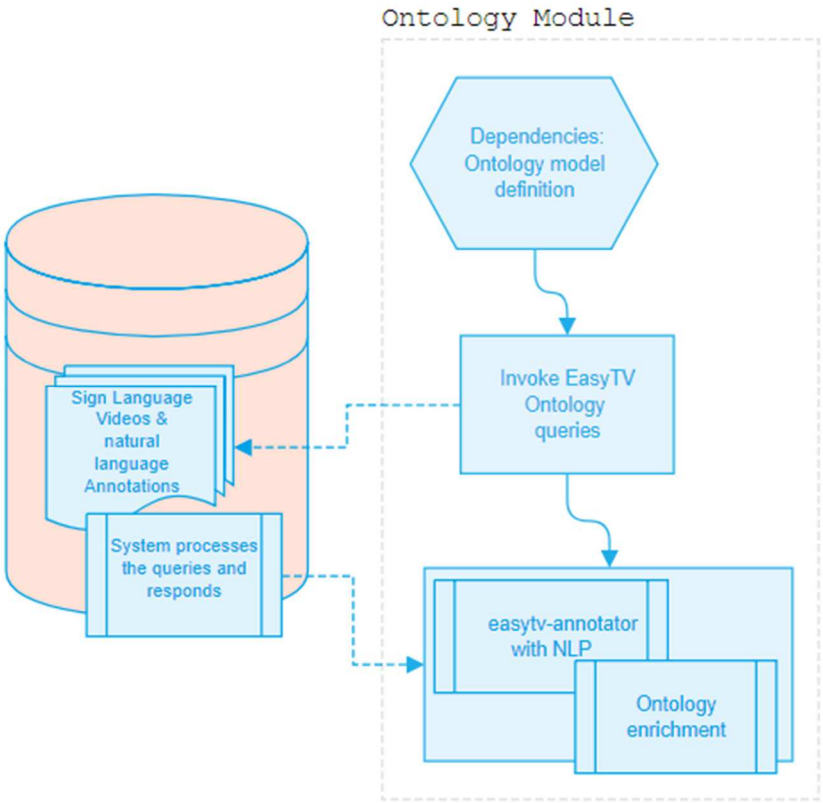


Figure 4: Scenario - Ontology module and platform interaction

3.3. Data models

In this section, we describe with natural language the data models that are employed from the EasyTV crowdsourcing platform. The scope of these data models is to define the main database entities that will support the creation, management and distribution of the crowdsourcing tasks. The description of the data models takes into account the communication requirements between the crowdsourcing platform and the EasyTV multilingual ontology module.

A user consists of the following entities:

- Full name

- Email address
- User's native language
- How user gets notified about new tasks or successful completion of current tasks
- Other languages that the user is proficient at
- Participation in projects (number of projects, completion status, etc.)

A project consists of the following data:

- The name of the project
- An informative description
- A project may include many tasks
- A project may have many participants
- A project may have many subscribers

A task is comprised of the following entities:

- The name of the task
- The type of the task (i.e., sign language or subtitle production)
- An informative description
- A task has submissions
- A task may have a confidence level

A submission consists of the following information:

- It may point to physical files, stored in the EasyTV repository
- It refers to one task
- It is submitted from a user
- It may be reviewed by many users

These data models are of paramount importance for the smooth operation of a crowdsourcing platform as they not only define important information that flows among the components of a crowdsourcing platform, but also define relationships among users, components, entities and functionalities of the platform.

4. FRONT-END AND BACK-END STACK

4.1. Frameworks

The survey for potential development frameworks was performed according to a number of criteria related to the challenges of the growing complexity that's involved in modern web development. For this purpose, popular community-backed projects are built atop of other open-source and well-tested modules. These frameworks incorporate and encourage the use of standardized software methodologies and depend on a number of community-backed tools that are continuously updated. For our purpose, we consider that an interactive platform even in prototype stage can benefit from such practices in a variety of ways, such as user experience, security updates etc.

In our case, it is desirable that the chosen framework is compatible with the concept of Hybrid Web Application – an application that combines a JSON API with server-rendered views. This means that in addition to an API, this type of application can serve dynamic HTML pages. Hybrid web apps can optionally employ a front end framework but not necessarily. For the scope of the EasyTV crowdsourcing platform, the chosen framework is expected to not only serve web interfaces for the human end users, but also expose APIs to facilitate the communication with other components of the EasyTV platform.

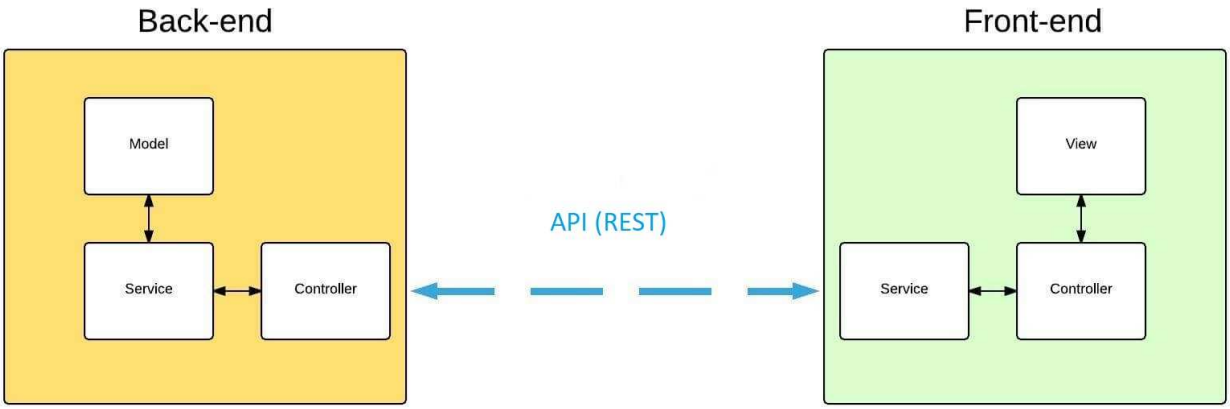


Figure 5: High level overview of Back-end – API – Front-end concepts

The block diagram in Figure 5 depicts the concept of *Back-end and Front-end decoupling* and provides a high-level abstraction about how data flows between the front-end and back-end.

To meet our needs, the Sails.js framework – a popular and actively developed micro-framework built on Node.js and its standard server, Express.js – is chosen because it offers the following advantages:

- It is written entirely on the Javascript programming language. This means that time spent on context-shifting is minimized and it is significantly easier for the developer to maintain the work's codebase.
- It is supported by CLI tools for code scaffolding.
- It follows the MVC pattern.
- It supports multiple databases with abstraction layer.
- Its data models get automatically a REST-API with CRUD operation.
- It provides basic security and role-based access control by default.
- It is agnostic to front-end choices.
- It is a general, extendable and pluggable structure.

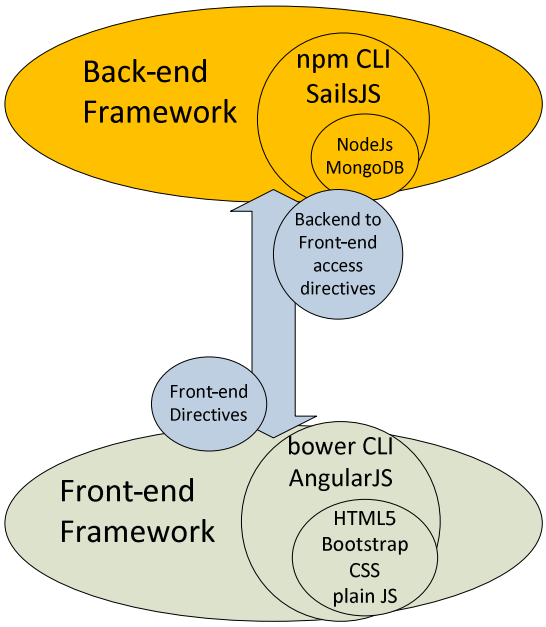


Figure 6: Development stack along development tools

In Figure 6 an overview of the main tools involved in the development of the EasyTV crowdsourcing platform is presented. These separate frameworks, apart from including automation tools for code scaffolding, they also encourage a number of guidelines for the development process.
As a front-end design solution, the crowdsourcing platform is based on Bootstrap[2], the most popular CSS framework. This framework was developed and open-sourced by Twitter in 2011 and can be installed in a project's front-end assets in the form of two development libraries, a CSS and a JavaScript. Bootstrap stands as a very popular choice both for initial prototyping and for further development as it offers a big collection of ready-to-use front-end components, which are responsive by default. Importantly, during the last years, the most recent versions of the library offer mobile-ready components, thus respecting the total share of mobile browsing globally. The components are used in a 'grid system' and more importantly they are easily customizable through CSS programming.

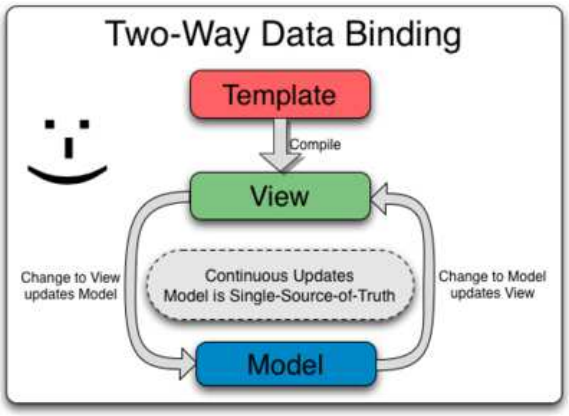


Figure 7: The concept of two-way data binding in Angular.js.

[2] https://getbootstrap.com/

Furthermore, in order to meet the need for dynamic web applications, Angular.js[3] framework is included in the front end development. This framework provides a toolset that its capabilities can be leveraged to create web applications with interactive UIs - dynamic views - instead of static server-rendered HTML. This capability is mainly possible through the Angular.js realization of the concept of two-way data binding, which allows for the components of the user-interfaces to be continuously synchronized with the updates of the actual data model. It is also important to note that through the Angular.js community, the developer has access to an ecosystem of reusable and extensible front-end components that are written in plain JavaScript and are supported by all modern browsers.

4.2. Mongo database

MongoDB[4] is a No-SQL database, in which data are stored in JSON-like hierarchical documents with key-value pair content. The employment of such a database for this phase of the development is justified by several reasons. Firstly, its No-SQL structure and dynamic schema fits for fast prototyping and allows keeping a simple design as the developer can modify the document's schema according to the data and not the data according to a strict schema. Secondly, a desirable trait of MongoDB is the fact that its engine is well supported within Sails.js framework database adapter as explained in its official documentation[5]. It provides an abstraction layer on top of the underlying database and this allows the development codebase to remain focused on the application's logic and bypasses the need for writing complex query language. Finally, considering the deployment and lifecycle of a web application, it is worth mentioning that an important feature of MongoDB is that it can be conveniently migrated to cloud infrastructure as various cloud services exist that support it[6].

4.3. Application's structure overview

The presented structure builds upon the platform's use scenarios and components' description. At this phase the overview focuses on the back-end functions that will serve the main data models and it also includes a reference to the access policy to these functions.

4.3.1. Outlining back end structure

In the following tables, we summarize the functionality of our back-end framework by presenting the basic endpoints grouped under the basic resource category. This presents a summary of the codebase structure and it outlines how various functionalities are grouped into the structure of the EasyTV crowdsourcing platform.

In general, the URIs description should follow a predictable, hierarchical structure to enhance understandability and usability[7].

Basic resource	**user/**	
Methods	GET	user/home
	GET	user/index
	GET	user/show?{id}
	GET	user/register

[3] https://angularjs.org

[4] https://www.mongodb.com/

[5] https://sailsjs.com/documentation/concepts/models-and-orm

[6] https://www.mongodb.com/cloud/stitch/

[7] https://www.restapitutorial.com/lessons/restfulresourcenaming.html

	PUT	user/signup?body:{name,email,language}
	POST	user/updateinfo?body:{name,email,language}
	POST	user/notify?{id}
	DELETE	user/destroy?{id}
Description	Controllers for managing user profiles	

Basic resource	**project/**	
Methods	GET	project/index
	GET	project/show?{id}
	GET	project/new
	POST	project/create?{body}
	PUT	project/edit?{body}
	POST	project/addmoderator?{id}
	GET	project/progress?{id}
	DELETE	project/destroy?{id}
Description	Controllers for managing projects. This includes basic CRUD operations and routing for displaying user interfaces	

Basic resource	**task/**	
Methods	GET	task/index
	GET	task/show?{id}
	GET	task/new
	GET	task/work?{id}
	POST	task/create?{body}
	PUT	task/edit?{body}
	DELETE	task/destroy?{id}
Description	Controllers for managing tasks. It includes basic CRUD operations and routing for displaying user interfaces	

Basic resource	**video/**	
Methods	GET	video/index?{project}
	POST	video/addtoproject?{body}
	POST	video/addannotation?{body}

	DELETE	video/destroy?{id}
Description	Controllers for managing videofile entities	

Basic resource	**file/**	
Methods	POST POST POST	file/uploadvideo?{body} file/uploadannotation?{body} file/uploadsubtitle?{id}
Description	Controllers for processing file streams	

Basic resource	**session/**	
Methods	POST DELETE	session/create/?{body} session/destroy?{id}
Description	Controllers for managing a user's session by authenticating	

Basic resource	**stats/**	
Methods	GET GET GET	stats/users?{id} stats/project?{id} stats/task?{id}
Description	Controllers for accessing functions related to statistics summaries and displaying the corresponding interfaces	

Basic resource	**settings/**	
Methods	GET POST POST	settings/index settings/update settings/reset
Description	Controllers for accessing administration settings	

4.3.2. **Defining access policies**

The following description about access policy implementation is based on the corresponding policy system employed by the development stack. The JSON-like script of Figure 8 presents an excerpt

of the Sails.js ACL (access control list) file which maps policies to the controllers in the back-end framework. The names of the policies for each endpoint refer to actual middleware functions in the sense that they form a layer of functions that is always executed before the actual controller is activated. These middleware functions may not be constrained to access-related logic. For instance, the 'localize' middleware function is always monitoring any change in the user's preferred language and applies it in the content of next response.

```
project: {
  'index': ['sessionAuth', 'moderator','localize'],
  'create': ['sessionAuth', 'moderator','localize'],
  'show': ['sessionAuth', 'moderator','localize'],
  'edit': ['sessionAuth', 'moderator', 'isModerator','localize'],
  'addmoderator': ['sessionAuth','moderator', 'isModerator', 'localize'],
  'destroy': ['sessionAuth', 'admin', 'localize'],
  'progress': ['sessionAuth', 'moderator','localize'],
  'new': ['sessionAuth', 'moderator','localize'],
},

settings: {
  '*': ['sessionAuth', 'admin', 'localize']
},
```

Figure 8: Policies and a corresponding policy controller defined in Sails.js

The flexibility in the definition of the middleware functions is depicted in Figure 9, in which the presented functions are applied when a user requests access to edit a project entity's information. The function on the left performs a simple check on the access rights of the logged user, which was specified upon his sign-in request. The function on the right is querying the 'Project' model in the database in order to confirm that the moderator belongs to the assigned moderators of the requested project.

```
module.exports = function(req, res, next) {

  // User is allowed, proceed to controller
  if (req.session.User && req.session.User.access === "moderator")
    return next();
  else if (req.session.User && req.session.User.access === "admin")
    return next();

  // User is not allowed
  else {
    FlashService.error(req, 'You must be a moderator.');

    req.session.returnTo = req.path;
    res.redirect('/');
    return;
  }
};
```

```
module.exports = function(req, res, next) {

  projId = req.param('id')
  // Moderator is owner, proceed to controller
  if (req.session.User && req.session.User.access === "moderator")
  {
    Project.findOne(projId)
    .populate('moderators')
    .exec( function(err, project) {
      if (req.session.User.id in project.moderators)
      return next();
    });
  }
  else if (req.session.User && req.session.User.access === "admin")
    return next();

    FlashService.error(req, 'You must belong to the moderators');

    req.session.returnTo = req.path;
    res.redirect('/');
    return;
  }
};
```

Figure 9: Middleware functions

4.4. File repository

A main functionality of the crowdsourcing platform is the handling, managing and storing of the crowd-users' submitted files. These files will be the answers submitted to the requested

crowdsourcing tasks and are going to be kept in a file structure that corresponds to the defined project-task hierarchy. Also the files will be related to each other through appropriate database fields in order for them to be accessed consistently through API requests.

4.5. Testing

The application of software testing tools is of high importance especially when a project is meant to be further developed and integrated with other modules in the future.

The Sails.js framework does not ship with a testing framework[8] neither explicitly suggests one. For the purposes of this development phase, we follow the community-proposed `Mocha` testing framework which targets the JavaScript language. The developer defines unit-test scenarios written in .js scripts which may include API requests or database queries in order to test the expected answers and the data integrity after the executed back-end operations. Depending on testing frameworks like Mocha is essential for handling the asynchronous nature of the back-end responses. For testing HTTP servers the framework can be integrated with tools like `supertest`[9].

```
var request = require('supertest');
should = require('should');

describe('UserController', function() {

  var Nusers;
  describe('#find_N_users()', function()
  {
    it('DB find function', function(done) {
      User.find().exec(function(err, users) {
        Nusers = users.length;
          done();
        })
    });
  });
  describe('#register user',function ()
  {
      var test_user = {
        firstName:'testUser',
        lastName:'testUser',
        email: "testmail@mail.com",
        email2: "testmail@mail.com",
        password: "testest",
        confirmation: "testest",
        userLanguage: 'English'
      };
      it('registers a new user',function (done) {
          request(sails.hooks.http.app)
            .post('/user/signup')
            .send(test_user)
            .expect(302, done);
      });
  });

  describe('#find_N+1_users()', function()
  {
    it('should equal N+1 users', function(done) {
      User.find().exec(function(err, users) {
          users.length.should.be.eql(Nusers+1);
          done();
        })
    });
  });

  describe('#login()', function() {
    it('should redirect to /', function(done) {
      request(sails.hooks.http.app)
        .post('/session/create')
        .send({
          username: "testmail@mail.com",
          password: 'testest'
        })
        .expect(302, done);
    });
  });
});
```

Figure 10. Mocha testing

A typical workflow may include the definition of an initialization procedure (launch the server and connect to database) and the execution of a number of HTTP requests that test the responses after each single step and inspect the integrity of the database. In Figure 10 it is presented a test file containing a flow of steps which test both the controllers and the database's state. The steps are

[8] https://sailsjs.com/documentation/concepts/testing

[9] https://github.com/visionmedia/supertest

given a short description in natural language. Initially the database is queried and a variable is set. The `supertest` library is used to execute a POST request with the appropriate arguments and the database is queried again to confirm the operation.

5. RELATIONSHIP WITH OTHER MODULES

In general, the functionality of the EasyTV crowdsourcing platform will depend on other EasyTV modules and components for the creation and visualization of the users' signs and on a next phase for the enrichment of multilingual ontology.

5.1.1. EasyTV capturing module

The communication between the platform and the **EasyTV capturing module** should handle a set of files containing both 3D motion data and user annotation information for the recorded signs. These files are meant to be uploaded to the crowdsourcing platform as a submission to a requested task, and stored into repositories in order to be available for other modules or end-users after their validation. Prior to the input validation, the moderator of the platform should have the ability to visualize the recorded RGB video and the corresponding annotation files. The inspection and validation step implies that the platform has already accepted the format of the submitted files during the task completion steps.

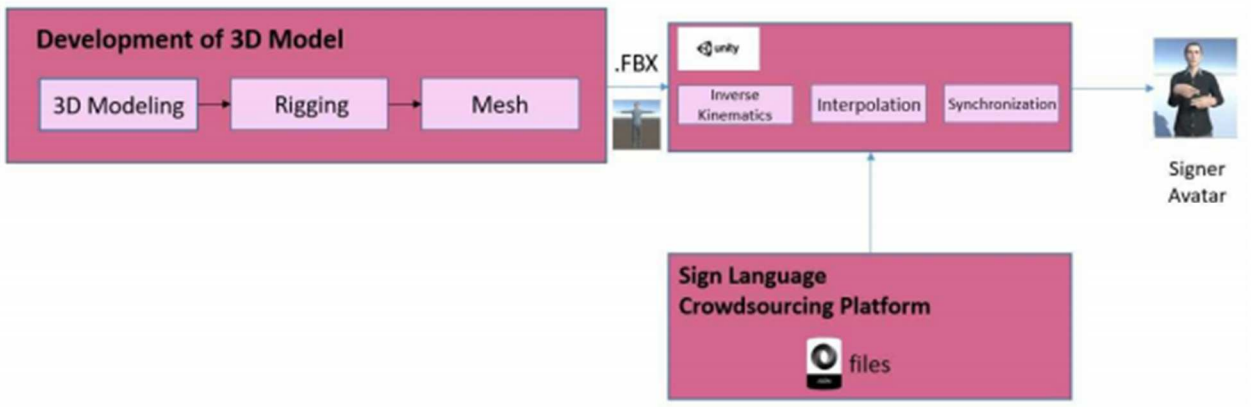


Figure 11: Schematic presentation of the input and output of the signer avatar service

5.1.2. Sign Language avatar playback

The work related to the creation of the realistic Sign Language avatar is described in the deliverable D2.1: "Sign language animation preliminary development and production" [15], which provides details about the tools and frameworks that are currently employed for avatar playback. The presented avatar development pipeline depends on the mocap data acquired through the completion of the crowdsourcing tasks using the EasyTV capturing module. One of the main purposes of the EasyTV crowdsourcing platform is to make the realistic avatar accessible, thus providing an EasyTV signer avatar service to the end-user.

During the preliminary development phase, the selected platform of the signer avatar creation team is Unity [16], a cross-platform development environment. Unity will power the main program for editing the avatar, connecting with crowdsourcing platform and translating JSON motion files into realistic graphics. The avatar service is meant to act as an external service thus will require access to the mocap data files of the repository through the network. Also the crowdsourcing platform is expected to be able to grant access to the avatar as a web service since Unity engine supports both Android and WebGL platforms.

5.1.3. Multilingual ontology module

The repository that will be created based on the crowdsourcing tasks should become available to the multilingual ontology service infrastructure. The development of the platform should expose an API that allows the ontology module to establish a two-way communication and exchange data with the platform.

According to the discussions made in [13] and [14], the preliminary versions of the EasyTV modules

employ a JSON file with a specific format. The EasyTV annotator library (Figure 12) receives from the crowdsourcing platform a Sign Language video with its natural language transcription sentence and its metadata.

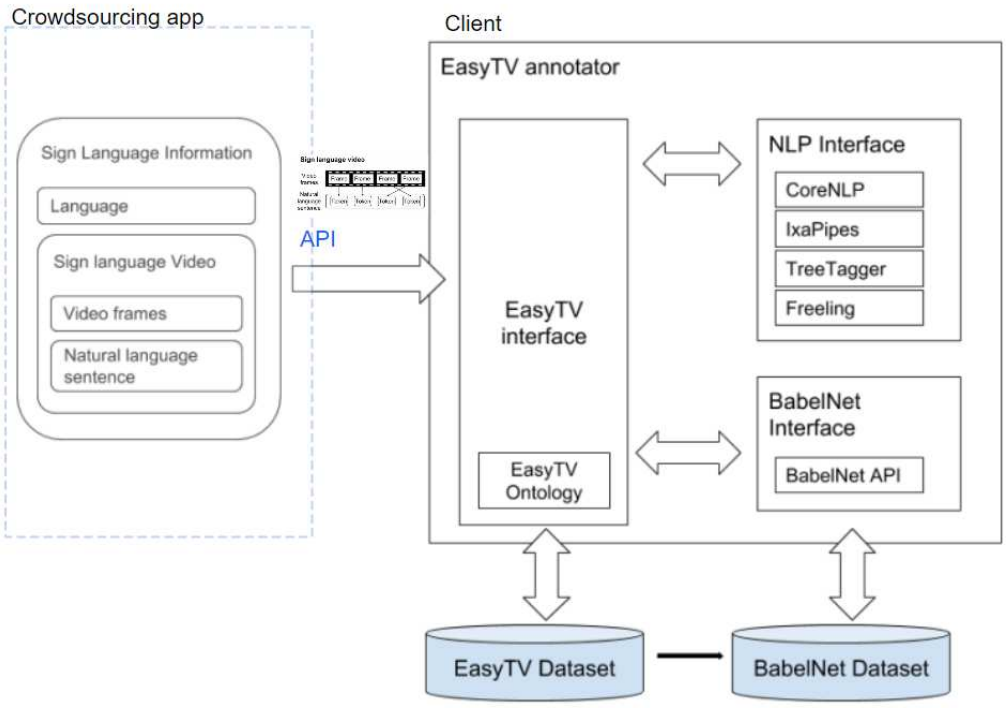


Figure 12: Crowdsourcing and the multilingual ontology

For this purpose, an external client should establish an authenticated session with the crowdsourcing platform prior to any requests to the API. During this first version of the platform, the authentication protocol of the client is based on the JWT[10] (JSON Web Token) open standard. In this standard the authentication workflow requires that the server sends a token to the requesting user that is used as an authentication credential in order to protect routes meant to be used by the client's application.

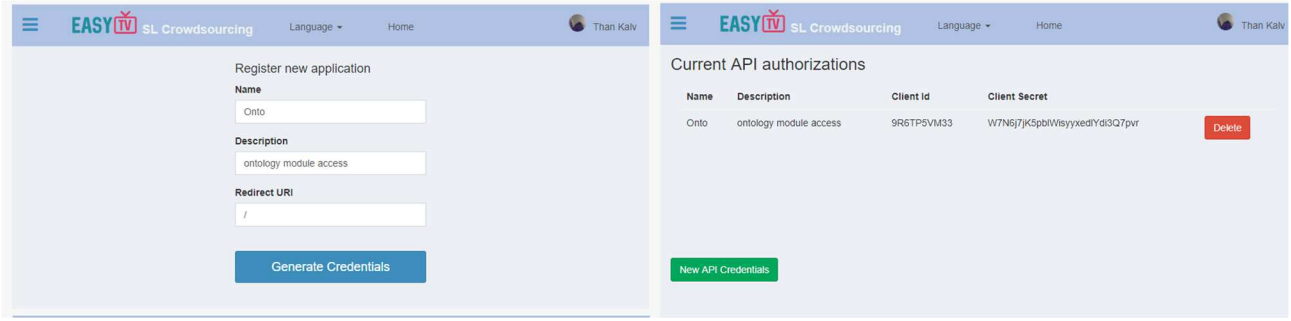


Figure 13: API key generation

Figure 13 demonstrates that prior to JWT authentication the client should have acquired API keys, with which the creation of a session will be possible. The client will perform the subsequent standard HTTP calls with JWT header authentication and the crowdsourcing platform will respond by providing a data structure. Subsequently, the client will use the appropriate paths to acquire the files, as shown

[10] https://jwt.io/introduction/

in Figure 14.

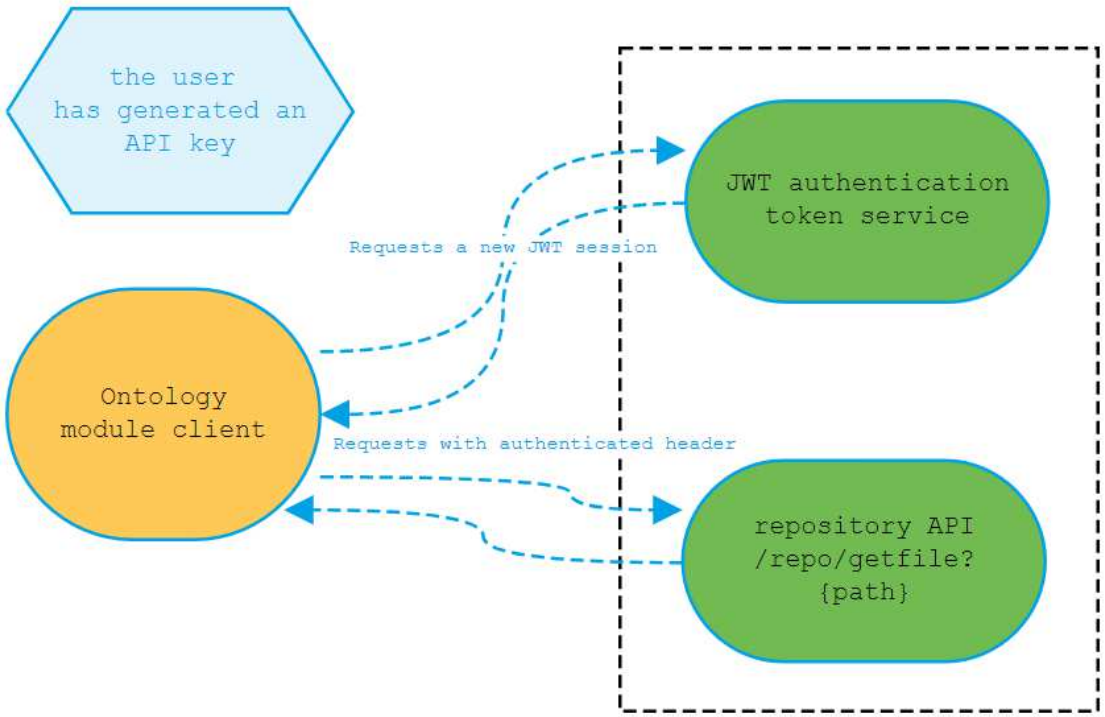


Figure 14: Client authenticated session

6. WORKFLOWS WITH GRAPHICAL INTERFACES

This chapter demonstrates different functionalities of the EasyTV crowdsourcing platform by providing descriptions and screenshots from the UI elements of the crowdsourcing platform.

6.1. User Signup and Registration

A user is able to sign up to the EasyTV crowdsourcing platform by providing a valid e-mail address and the corresponding correct password as shown in Figure 15. Furthermore, a user that is not currently registered to the platform can easily register by providing his/her first and last names, language and e-mail address, as shown in Figure 16.

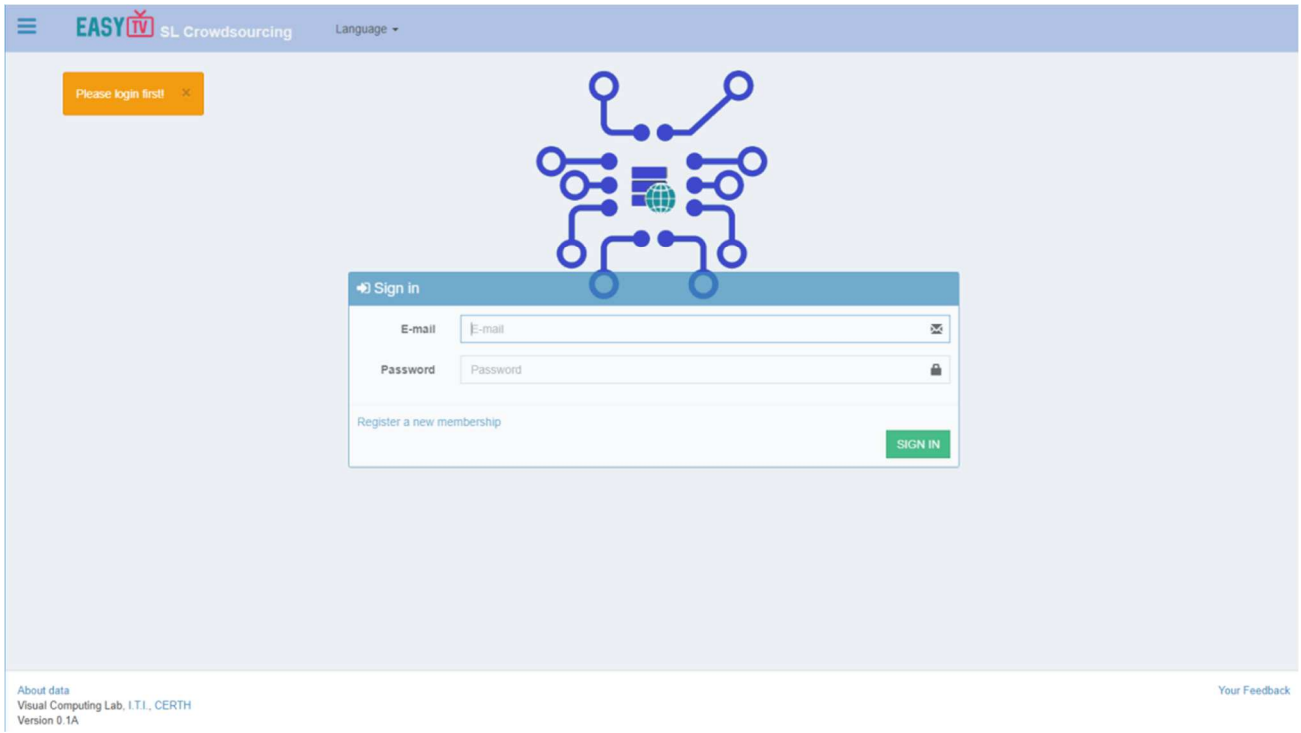


Figure 15: The user login screen appears whenever a user wants to connect with the EasyTV crowdsourcing platform.

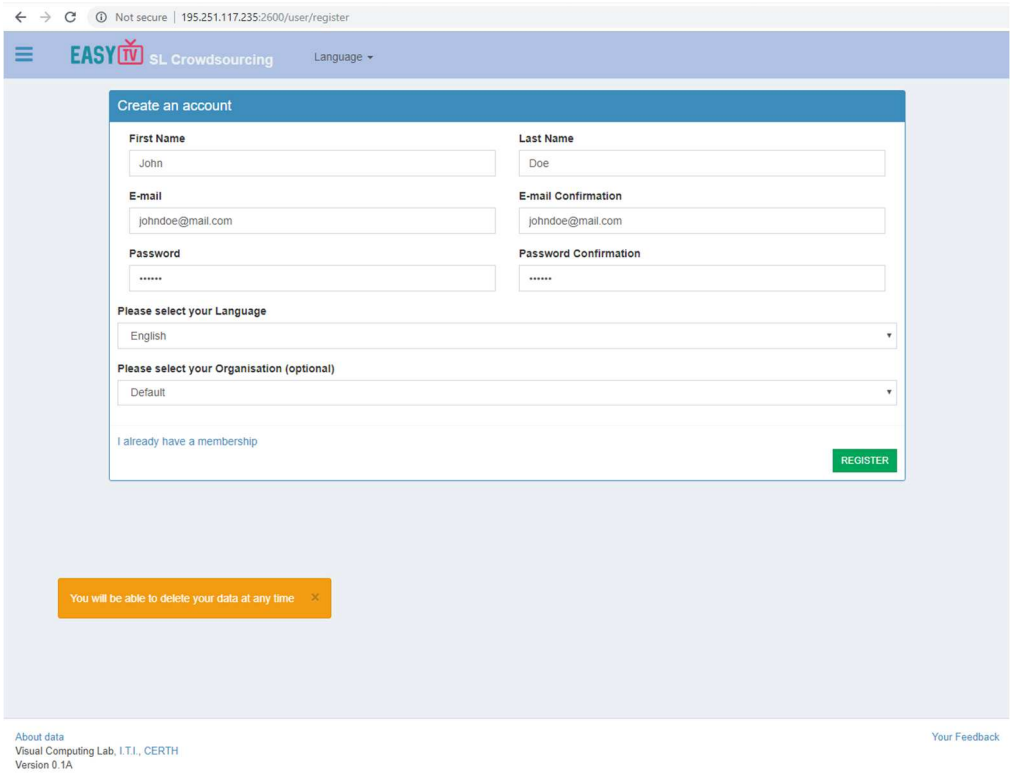


Figure 16: A user can easily register to the EasyTV crowdsourcing platform by filling the above form.

6.2. Defining a project and distributing tasks

In this section, we describe the functionality of the crowdsourcing platform that enables the creation of new projects and the creation and distribution of tasks for the projects. Initially, we present the definition of project and task, as well as the required fields that need to be completed before we describe in detail the functionality of project and task creation.

- *Project* – a collection of tasks with a defined goal
- *Task* – the sequential steps of a job
    i) Language options
    ii) Expected number of contributions and users (profiles' requirement)
    iii) Verification method

6.2.1. Workflow of the user-moderator

The scope of the *project-creator-manager* tool is to provide users with the ability to create and edit platform-related content through intuitive interfaces. This has been the main focus of the development for the first prototype of the EasyTV crowdsourcing platform.

A task workflow can be summarized in four main steps:

- Provide a tool for creating projects and defining tasks
- Communicate (broadcast) the project tasks to a pool of users
- Accept and process user's input in the system
- Validate the content and update the project status

In the framework of the EasyTV crowdsourcing platform, a new project is created by defining a name, language and category and by providing information that end-users may find useful in order to understand the project and successfully complete it (shown in Figure 17).

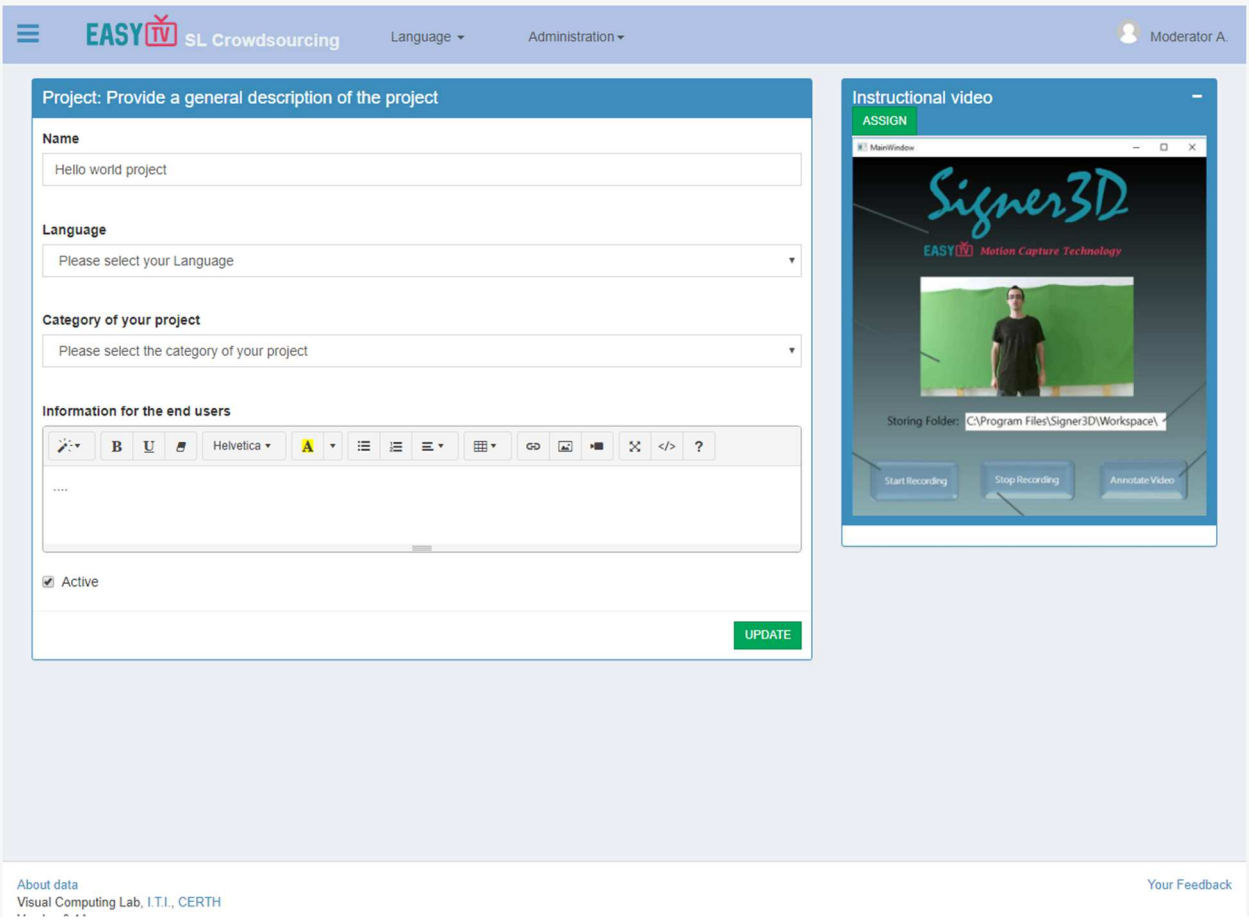


Figure 17: Creation of a new project by providing important details and information about the project.

As far as the task creation is concerned, a moderator can define a new task for a given project by filling the form, shown in Figure 18, with all the necessary information that include the name and type of the task, the method used for the verification of the task completion and important information that can assist the interested users in successfully completing the task.

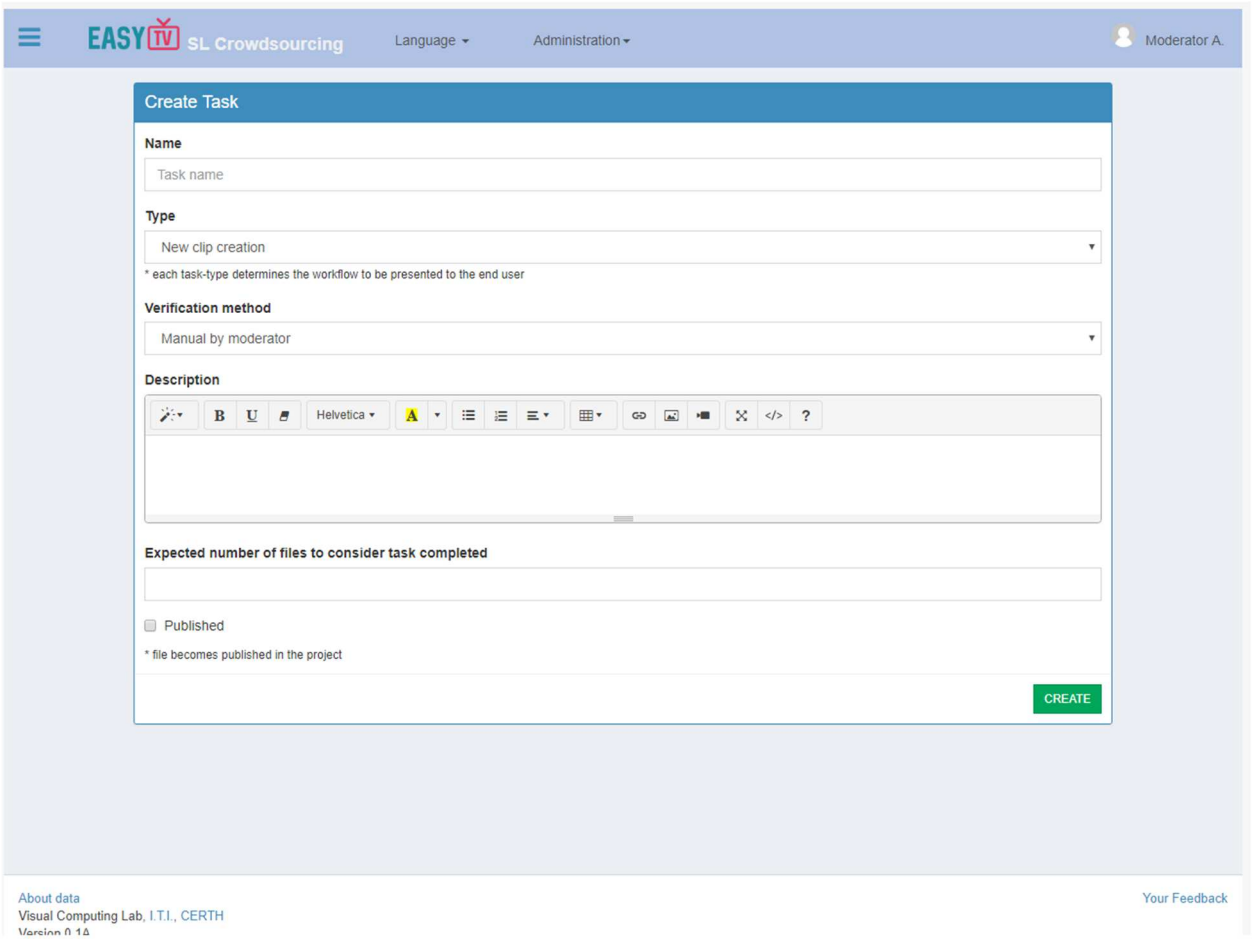


Figure 18: Definition of a new task by completing the requested information.

Furthermore, the moderator is given the option to notify specific users about the presence of a new task (see Figure 19). This notification is performed by offsite communication methods and has as a purpose to disseminate faster the news about the presence of new tasks and, as a result, achieve a possibly faster response and task completion time. Furthermore, the moderator can inform end-users that have a higher trust level and therefore achieve higher quality task completion results.

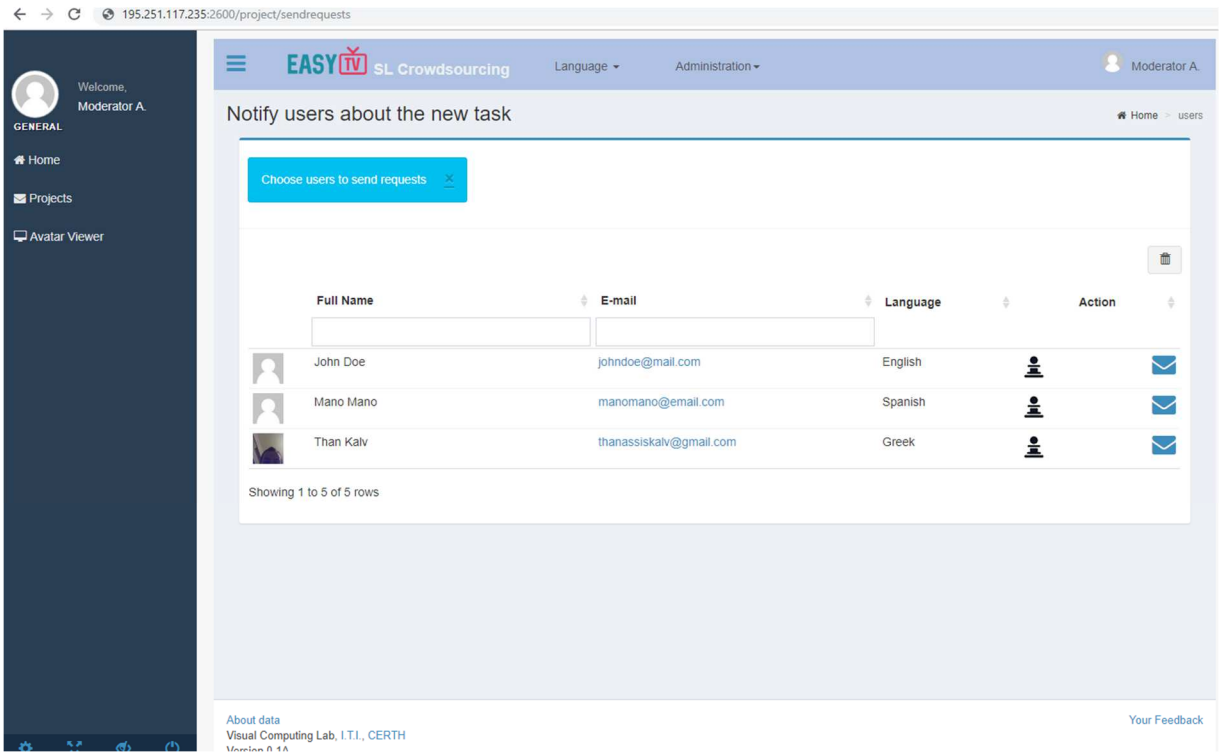


Figure 19: The moderator is presented with a user notification screen, from which he/she can notify specific users about the presence of new task that needs to be completed.

6.2.2. Workflow of the user-worker

This section describes the procedure that a user-worker should follow in order to complete a specific active task. The procedure for the task completion is presented as a workflow in the EasyTV crowdsourcing platform and it consists of a number of **steps in wizard-style interface,** which prompt the user to provide the necessary inputs in order to successfully complete the task. It is crucial to allow for a simple and easy navigation between the steps in order to facilitate the tasks of a user-worker.

The steps required for the task completion by a user-worker are described below, along with illustrations of the UI elements of the crowdsourcing platform. Initially, the user is presented with general information about the task and the steps that he/she should follow to successfully complete it (Figure 20 (top)). Afterwards, the user can upload the necessary video and motion files that correspond to the correct task (Figure 20 (bottom)). These files are generated by employing the Sign Language Capturing module that was described in detail in deliverable D3.1: “Sign language capturing technology preliminary version” [14]. The Sign Language Capturing module enables the recording of video files and the extraction of necessary features that will facilitate the sign language recognition and avatar playback functionalities.

Afterwards, the user should upload the annotation file that is usually a text file that contains the word or set of words (with correct order) that the user signs in the uploaded video file (Figure 21 (top)). Finally, the EasyTV crowdsourcing platform informs the user about the successful upload of all the required files, accepts the submission and informs the corresponding moderator about the presence of a new submission, which receives the status of “under review” (Figure 21 (bottom)).

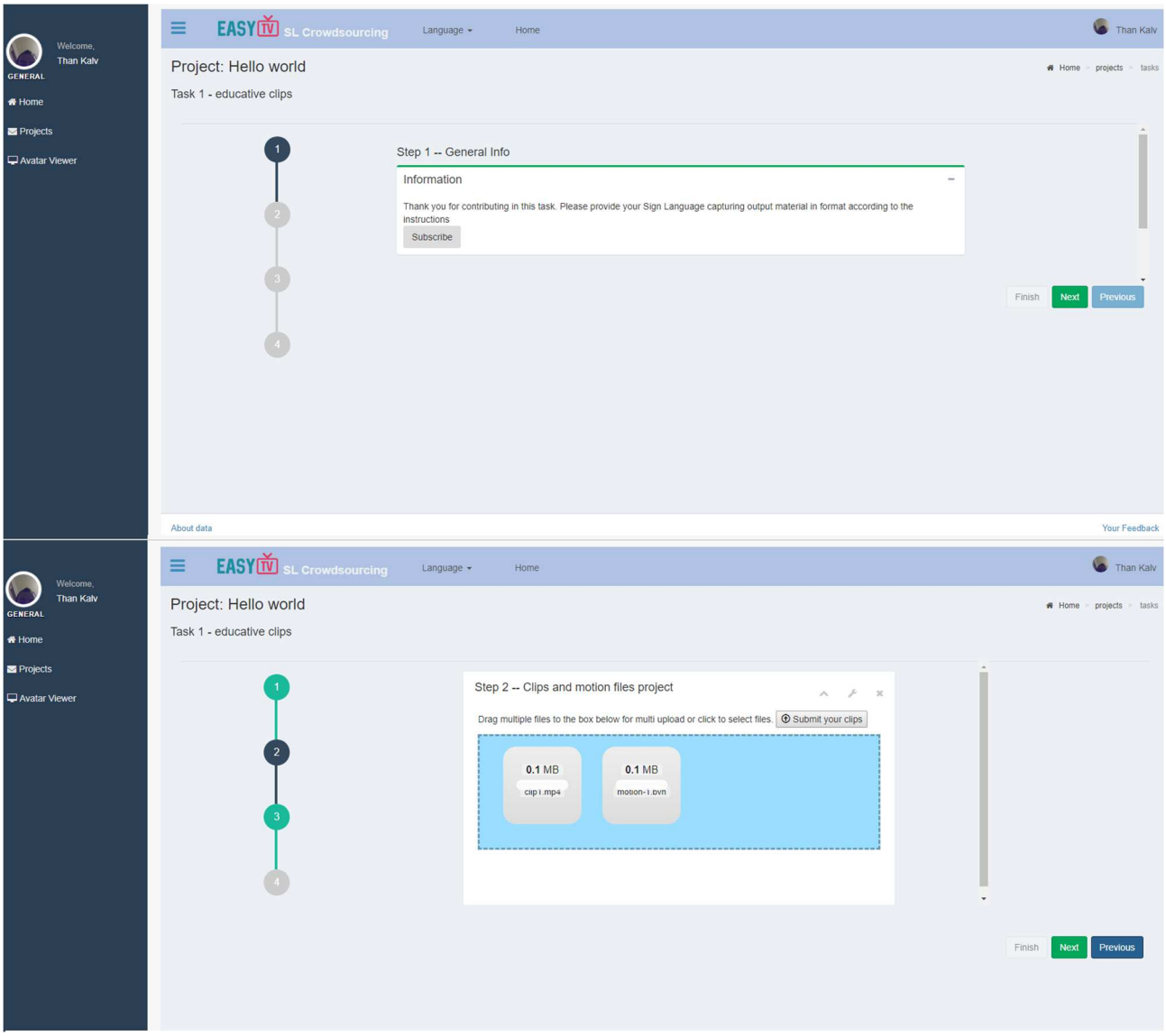


Figure 20: "Wizard" steps of a task workflow describing the general information presentation (top) and the upload of the video and motion files (bottom).

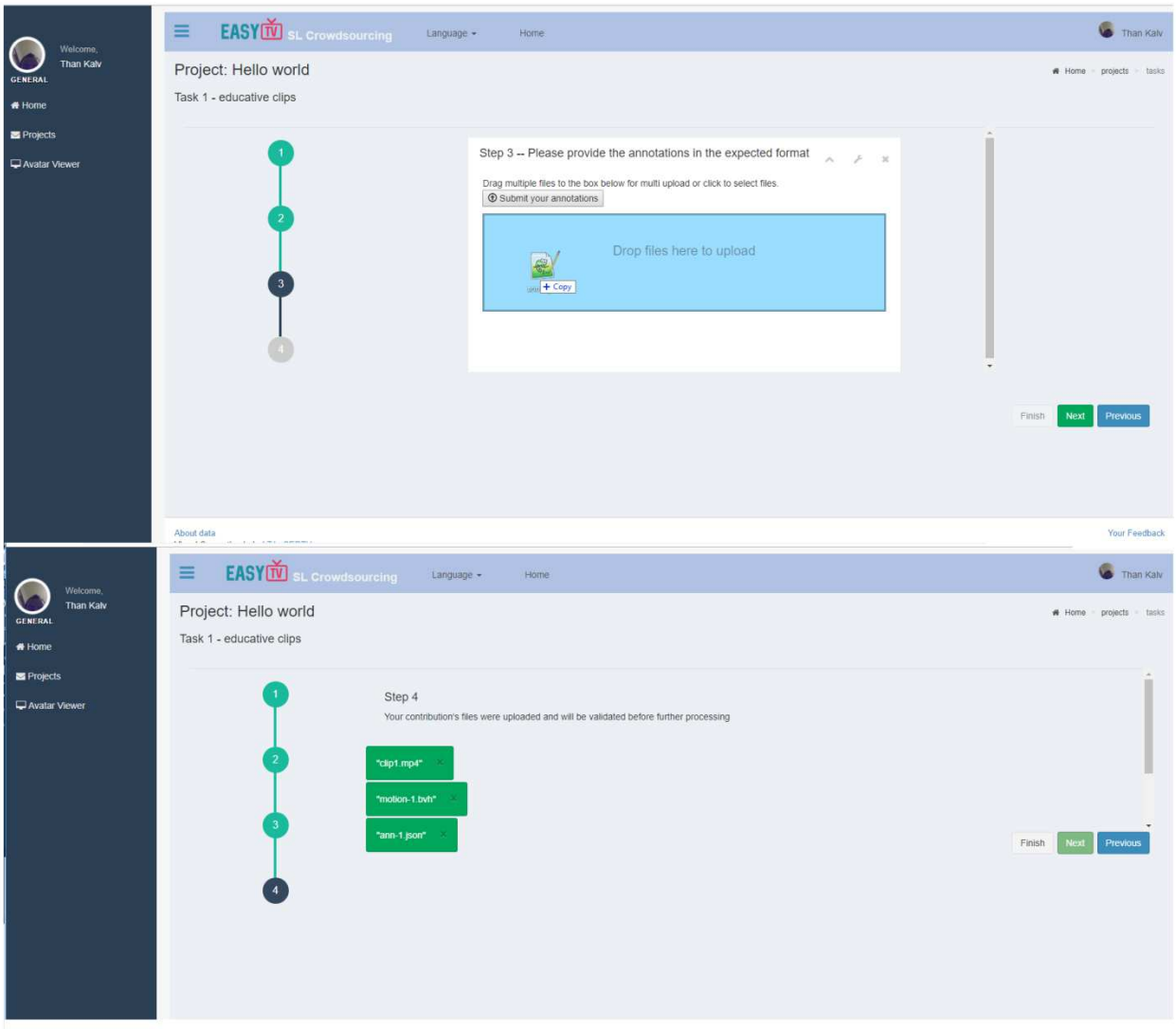


Figure 21: "Wizard" steps for the task workflow describing the upload of the annotation file (top) and the notification of the user about the correct submission of the required files (bottom).

6.3. Task assessment and feedback

In this section, we describe the procedure required so that a moderator accepts or rejects a new submission. Once a moderator is informed about the presence of a new submission, he/she is able to access the submission and view the submitted material (Figure 22). The moderator is then able to validate the material based on his/her knowledge of the problem and accept or reject the submission. The decision of the moderator is also passed to the user, who is notified about the result. In the case of acceptance, a final version of the submission is stored on a data repository and is available in the Data Storage Component. The crowdsourcing platform also enables the moderator to be informed about the current progress of the project and realize how close it is to completion.

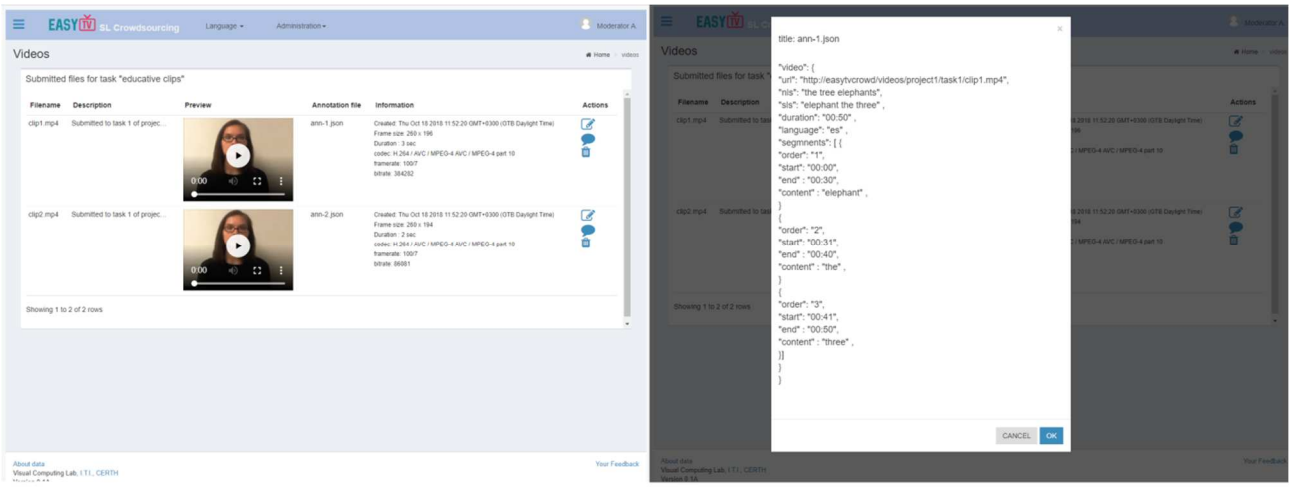


Figure 22: Assessment of the uploaded material by the moderator.

7. CONCLUSION AND FUTURE WORK

Throughout this document the focus has been on the architecture and the employed software infrastructure of the EasyTV crowdsourcing platform. The involved concepts were documented with flowcharts, mock-ups and excerpts of the source code of the development structure. The referenced technical details present the development stack that was selected and how it is involved in the design and development process of the crowdsourcing web application.

The platform's status is currently in an intermediated version, aspects of which will be continuously updated and upgraded in the coming months along with the development of the rest EasyTV components. The user interfaces presented should be considered as a prototype version and it should expected that in following versions they will be enriched with more elements and dynamic components.

It is important to note that as the data repository will gradually be populated with real data, its structure and its database will naturally be under consideration and may be reconfigured upon feedback from the other interested EasyTV modules.

The expected development steps can be summarized:

- Refinement of the prototype UIs presented in this stage
- Refinement of the API for the interested clients
- Perform technical testing towards integration
- Management of security issues
- Consider the needs of scaling the application

8. REFERENCES

[1] Brabham, D. C., "Crowdsourcing as a model for problem solving: An introduction and cases," *Convergence*, *14*(1), pp. 75-90, 2008.

[2] Howe, J., "The Rise of Crowdsourcing," *Wired Magazine*, June 2006.

[3] Saxton, G.D., Oh, O. and Kishore, R., "Rules of Crowdsourcing: Models, Issues, and Systems of Control," *Inform Syst Manage*, vol. 30, no. 1, pp. 2–20, 2013.

[4] Hoßfeld, T. et al. "Survey of web-based crowdsourcing frameworks for subjective quality assessment," *IEEE 16th International Workshop on Multimedia Signal Processing (MMSP)*, Jakarta, pp. 1-6, 2014.

[5] Buettner, R. "A Systematic Literature Review of Crowdsourcing Research from a Human Resource Management Perspective," *48th Annual Hawaii International Conference on System Sciences*, Kauai, Hawaii: IEEE. pp. 4609–4618, 2015.

[6] Aitamurto, T., "Crowdsourcing as a Knowledge-Search Method in Digital Journalism: Ruptured Ideals and Blended Responsibility," *Digital Journalism*, vol. 4, pp. 280–297, 2016.

[7] Taeihagh, A., "Crowdsourcing: a new tool for policy-making?," *Policy Sciences*, 50 (4), pp. 629–647, 2017.

[8] Créquit, P., "Mapping of Crowdsourcing in Health: Systematic Review", *Journal of Medical Internet Research*, 20 (5), 2018.

[9] Memrise – Learning, made joyful: https://www.memrise.com/ (Accessed on 18/10/18).

[10] D1.4: "Final release of the EasyTV system architecture (https://easytvproject.eu)."

[11] D1.1: "User scenario and requirements definition."

[12] D1.2: "EasyTV system requirements specification."

[13] D3.2.1: "Enriched multilingual ontology with signs in different languages preliminary version."

[14] D3.1: "Sign language capturing technology preliminary version".

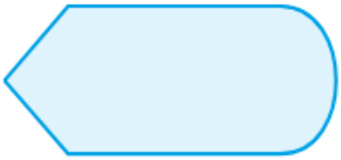
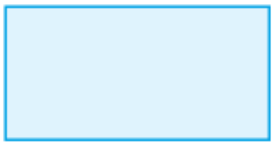
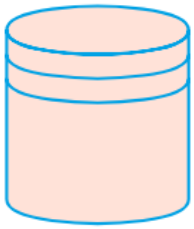
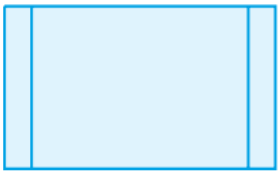
[15] D2.1: "Sign language animation preliminary development and production".

[16] Unity3D,"Interpolation."[Online].Available:https://docs.unity3d.com/ScriptReference/Rigidbody-interpolation.html (Accessed on 30/09/18).

APPENDICES

1. Flowchart shapes for user scenarios

In this section, we present the links between flowchart shapes and their semantic relations. These shapes are employed in order to clearly illustrate the user roles and functionalities in developed EasyTV Crowdsourcing Platform presented in the current deliverable.

 **Preparation**	 **Display**	 **Process**
 **Database**	 **Manual User Input**	 **Database documents**
 **Subprocess**	 **Workflow direction**	 **Communication**

2. Presentation of basic use cases with interfaces

In this section, we describe three possible scenarios that clearly illustrate the interaction between the users and the EasyTV crowdsourcing platform. In a nutshell, the first scenario describes the interaction between a user-worker with the platform and it involves the procedures of logging in the platform and completing a task. The second scenario describes the procedures, in which a user-moderator creates a project and tasks that can be distributed for completion by users, while the third scenario illustrated the procedures involved in enabling a user-moderator to review a submission and decide on the successful completion of a task.

User logins and completes a task

This scenario describes the interaction of a signer worker with the EasyTV crowdsourcing platform, while he/she attempts to login to the platform and accomplish a task. This scenario also involves the interaction of the user with another important module of the EasyTV system architecture, which is the Sign Language Capturing module that was introduced and thoroughly explained in deliverable D3.1: "Sign language capturing technology preliminary version" [14]. The Sign Language Capturing module is necessary as it generates the necessary files that the user should upload to the crowdsourcing platform in order to successfully complete the task that was assigned to him/her. This scenario is described in detail below, along with screenshots of the corresponding UI elements of the EasyTV crowdsourcing platform.

STEP 1: The user authenticates to the EasyTV crowdsourcing platform by using his/her e-mail and password.

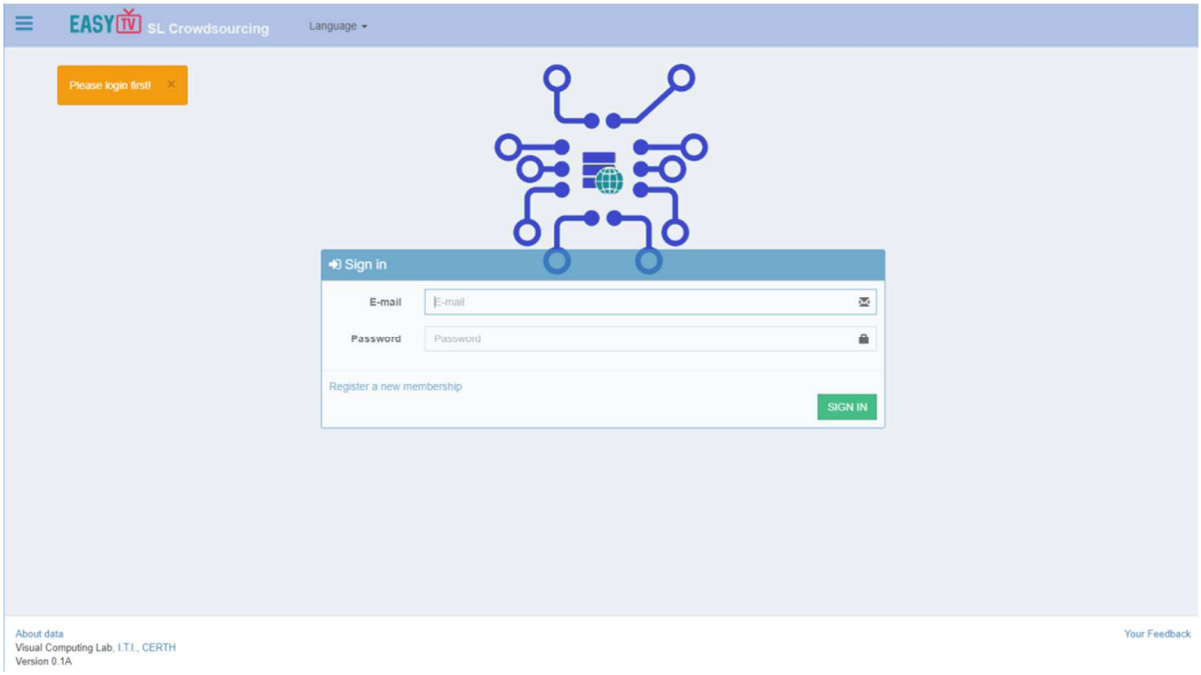


Figure 23: User login in the EasyTV crowdsourcing platform.

STEP 2: The user is presented with the active projects and relative tasks. Tasks can then be selected and completed by the user.

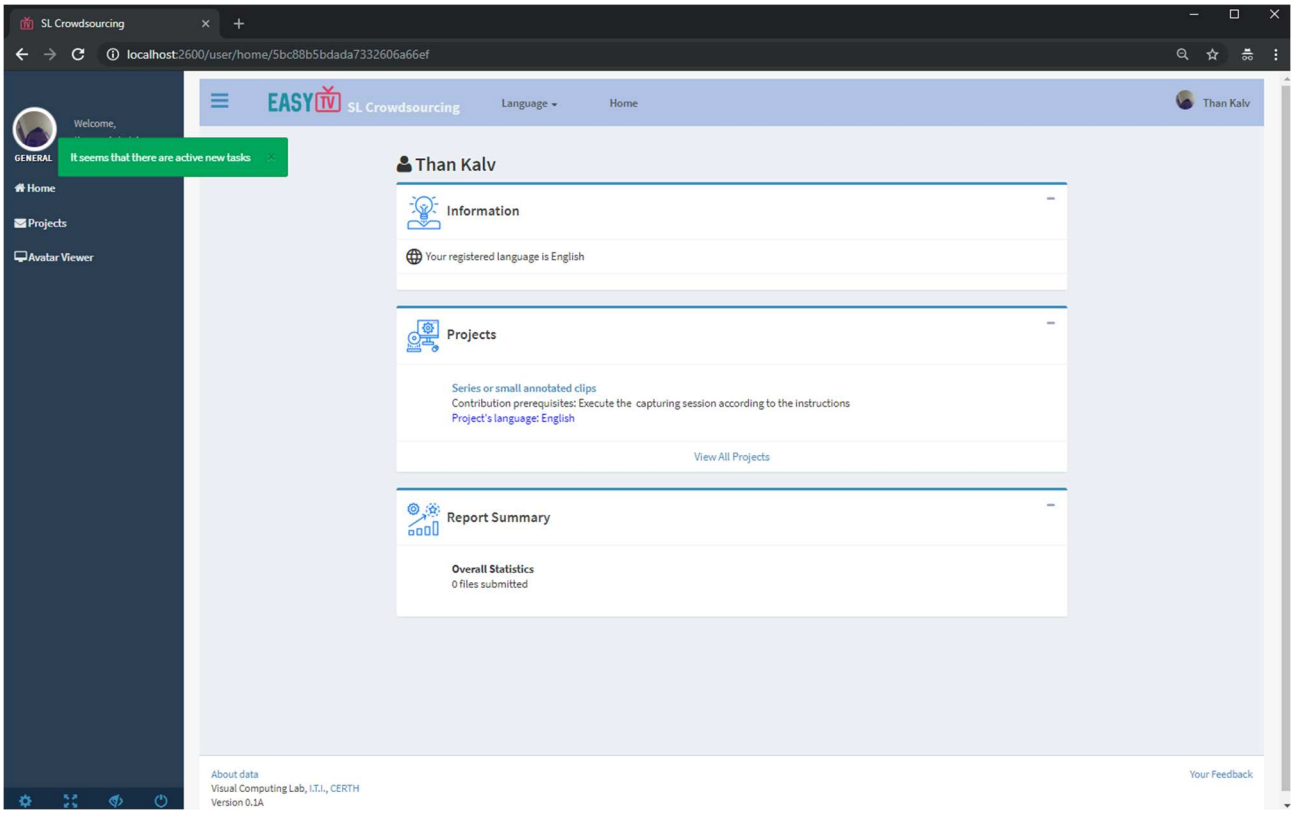


Figure 24: The home page of active tasks that the user can select and complete.

STEP 3: The user is presented with a workflow of the task to complete. The workflow task consists of a series of steps that the user should execute successfully in order to complete the task itself.

a) The user initially reads the details of the task.

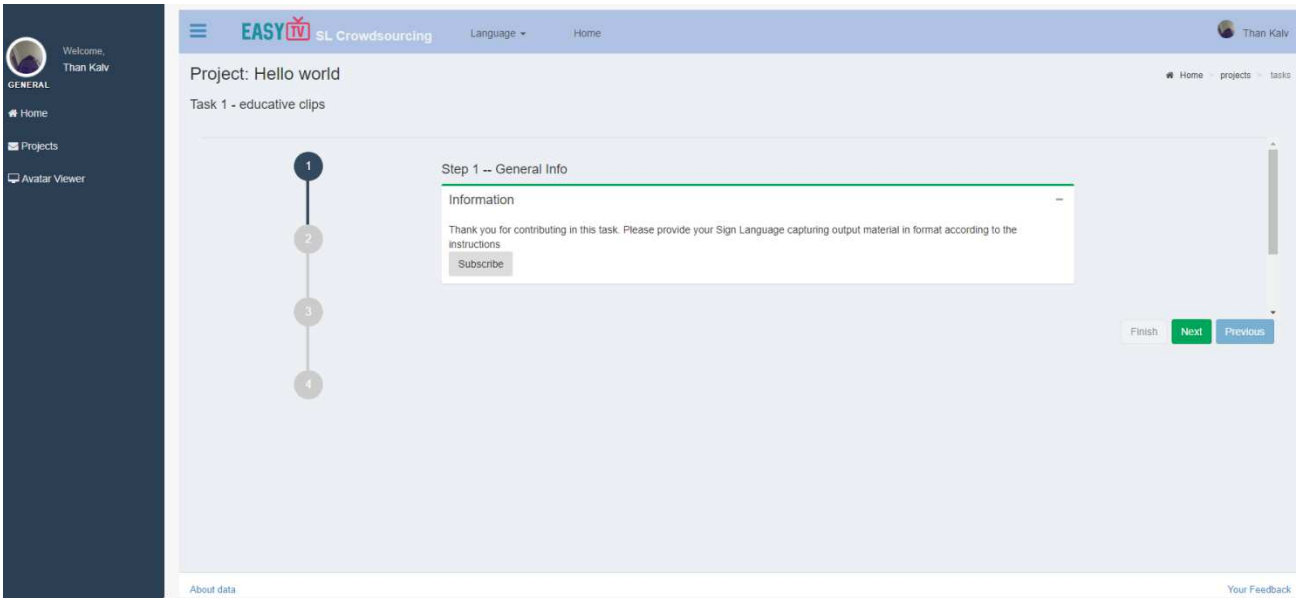


Figure 25: General info about the task and the procedure required to complete it are initially presented.

a) Afterwards, the user uploads video and motion files related to the task.. These files are produced as output from the Sign Language Capturing module as it was previously described.

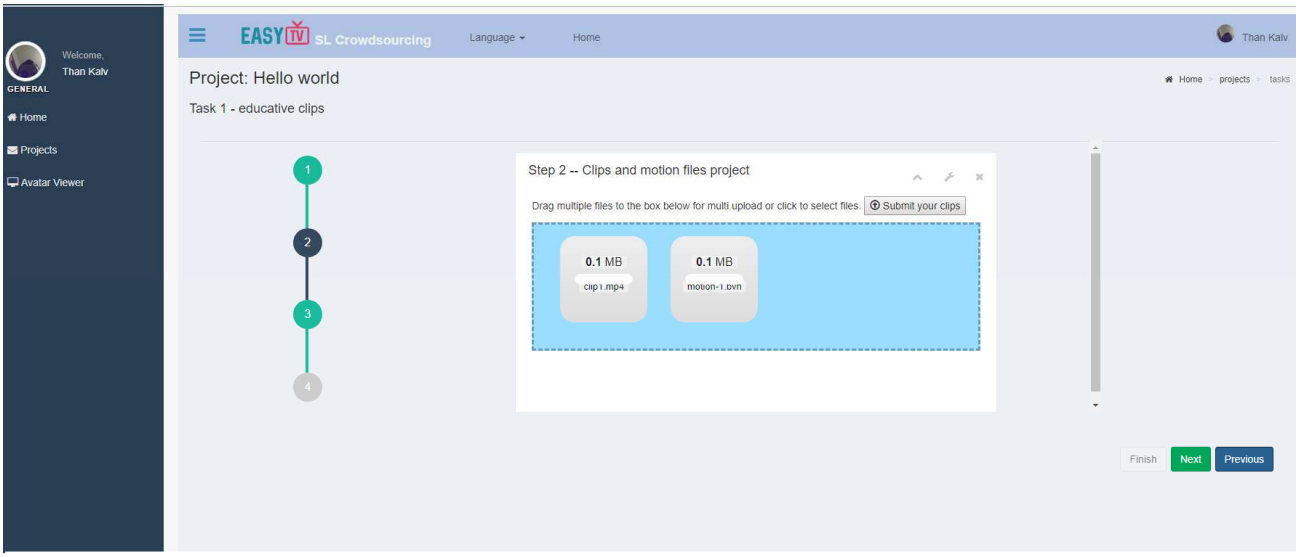


Figure 26: The user can upload video and motion files to the crowdsourcing platform.

c) Then, the user uploads an annotation file that describes the task. In the EasyTV sign language framework, the annotation file can be a text file that contains the word or set of words that are signed on the video file that was uploaded in the previous step.

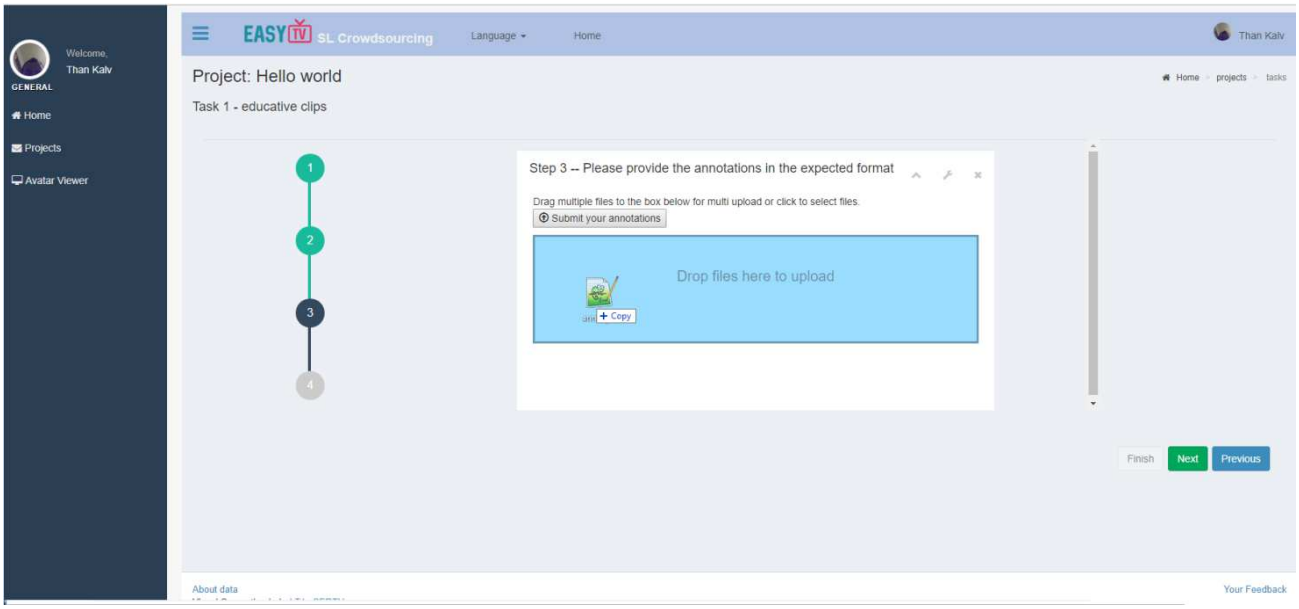


Figure 27: The user uploads an annotation file to the EasyTV crowdsourcing platform.

c) The EasyTV crowdsourcing platform confirms that it received the requested files and accepts the submission by the user. Then, the platform sets the status of the task as “under review” so that a moderator is notified to review the task and either accept or reject it.

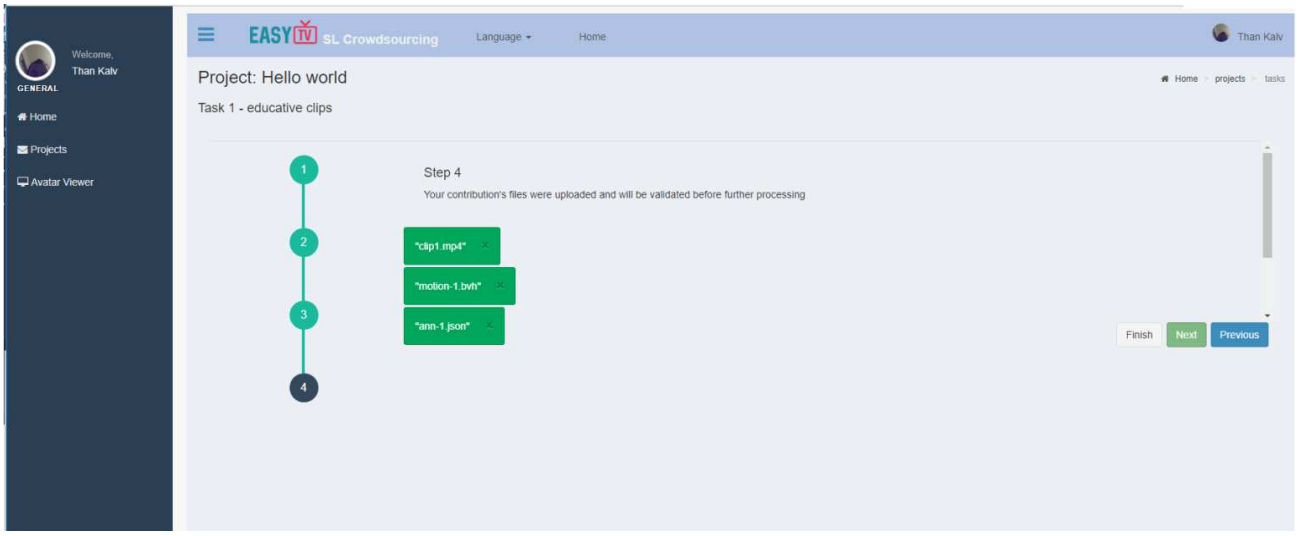


Figure 28: The EasyTV crowdsourcing platform confirms that it received the reqeusted files and accepts the submission.

Moderator creates a new project and a new task

This scenario describes the interaction of a moderator with the EasyTV crowdsourcing platform. In this scenario, the moderator exploits his/her ability to create a new project and tasks that can be distributed to users for completion. In the following steps, we describe and illustrate with screenshots from the corresponding UI elements of the crowdsourcing platform, the procedure required by the moderator in order to create a new project.

STEP 1: The user-moderator is presented with a login screen, where is asked to enter a valid e-mail address and password in order to connect with the EasyTV crowdsourcing platform.

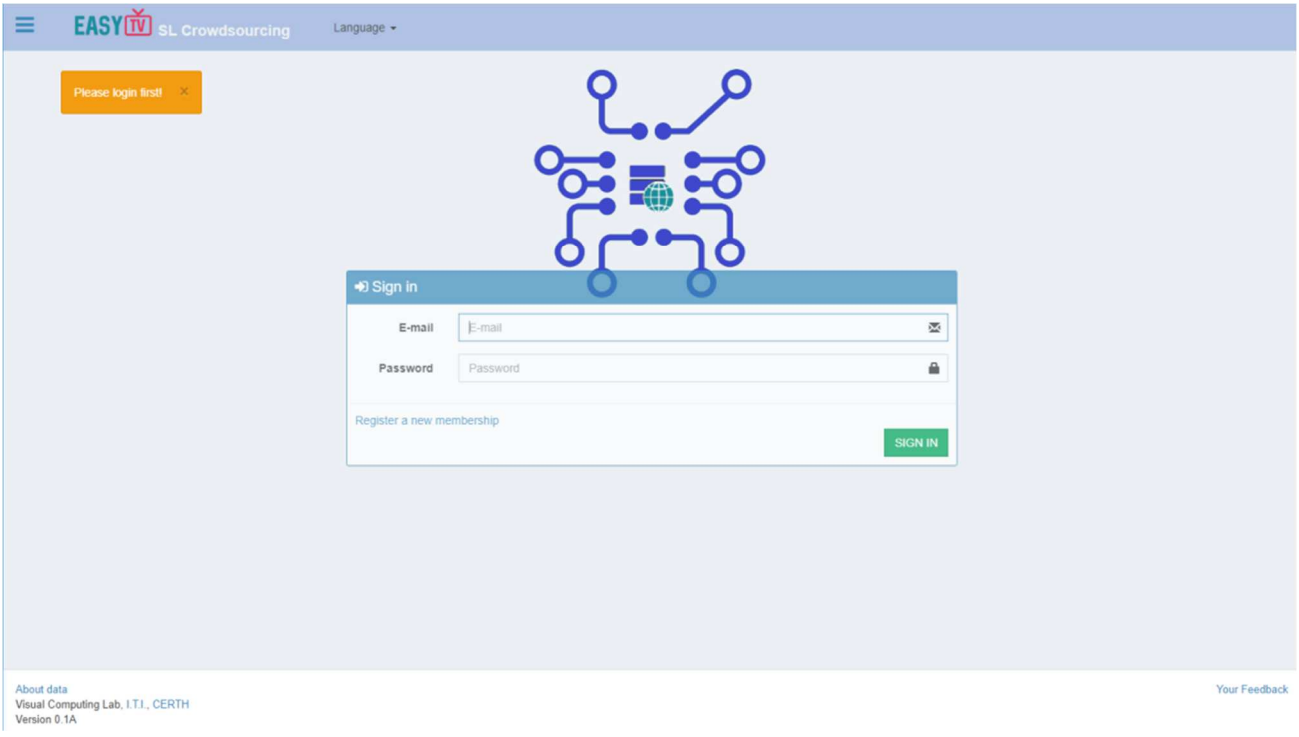


Figure 29: The moderator is presented with a login screen, when he/she attempts to connect with the EasyTV crowdsourcing platform.

STEP 2: The user-moderator launches a new project by setting a name and some important information or instructions that can assist users in order to successfully complete it.

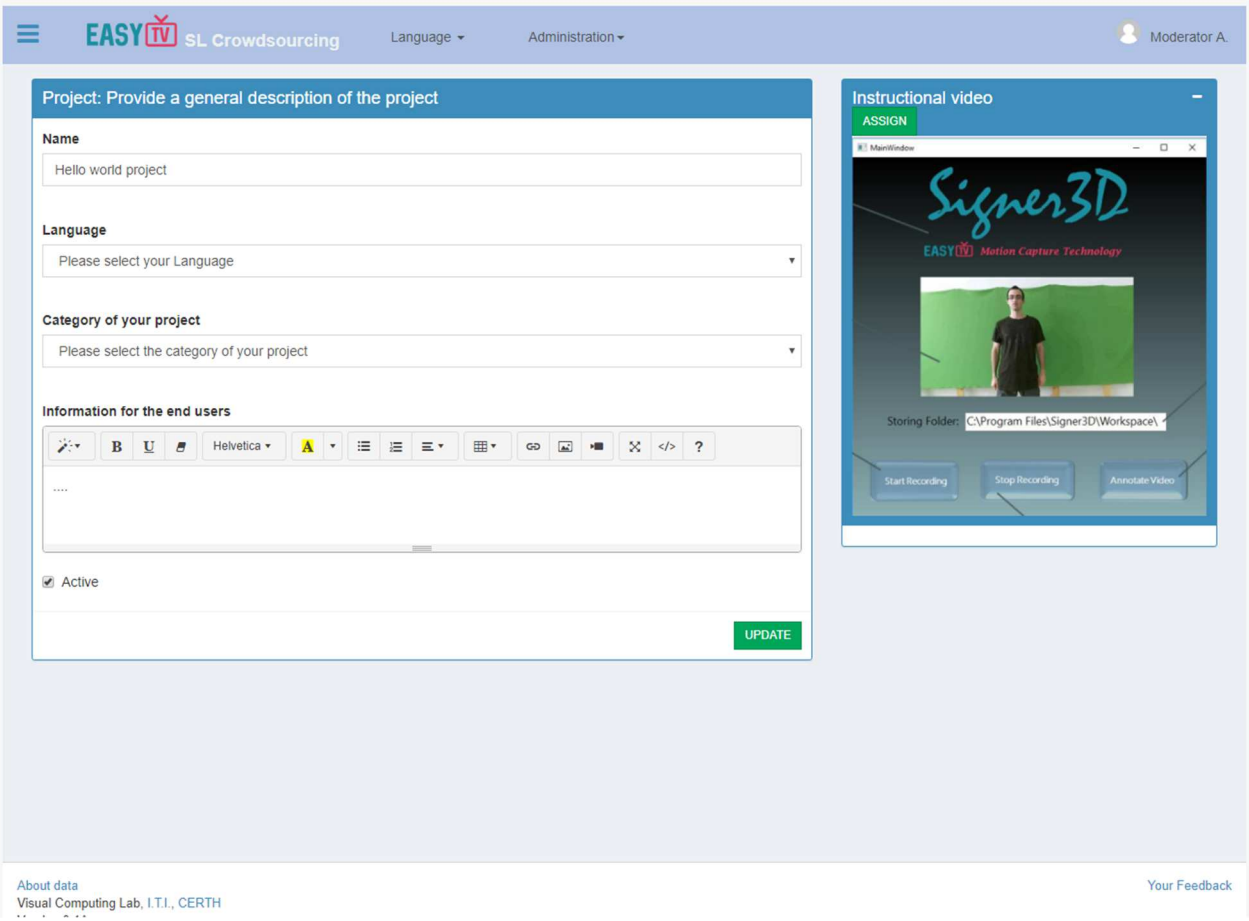


Figure 30: The moderator sets the name and important information for the new project he/she want to launch.

STEP 3: The user-moderator defines a new task for the launched project by typing a name and description. The new task is then added to the pool of active tasks, from which a user can draw and complete.

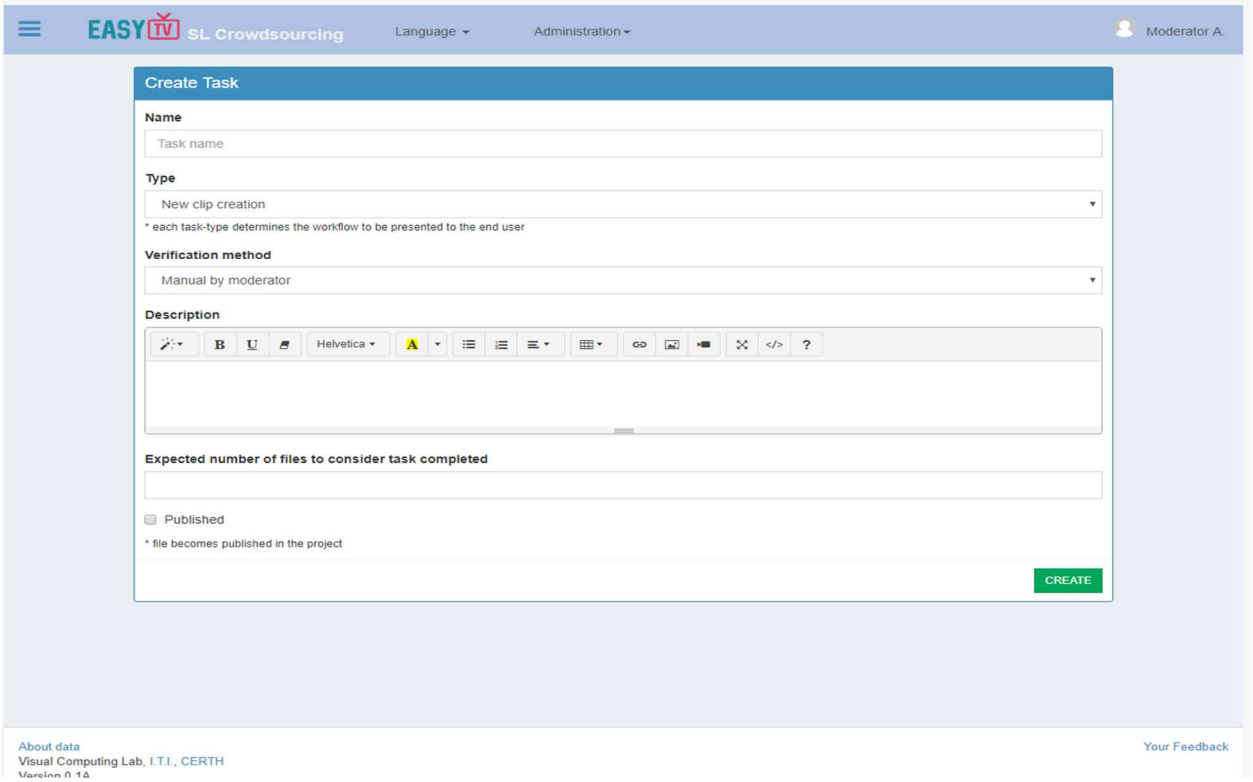


Figure 31: The moderator creates a new task for a launched project by typing a name and description.

STEP 4: The user-moderator notifies potential workers about the existence of a new task by using an offsite communication method. This feature enables the faster dissemination of the new tasks and therefore their faster completion.

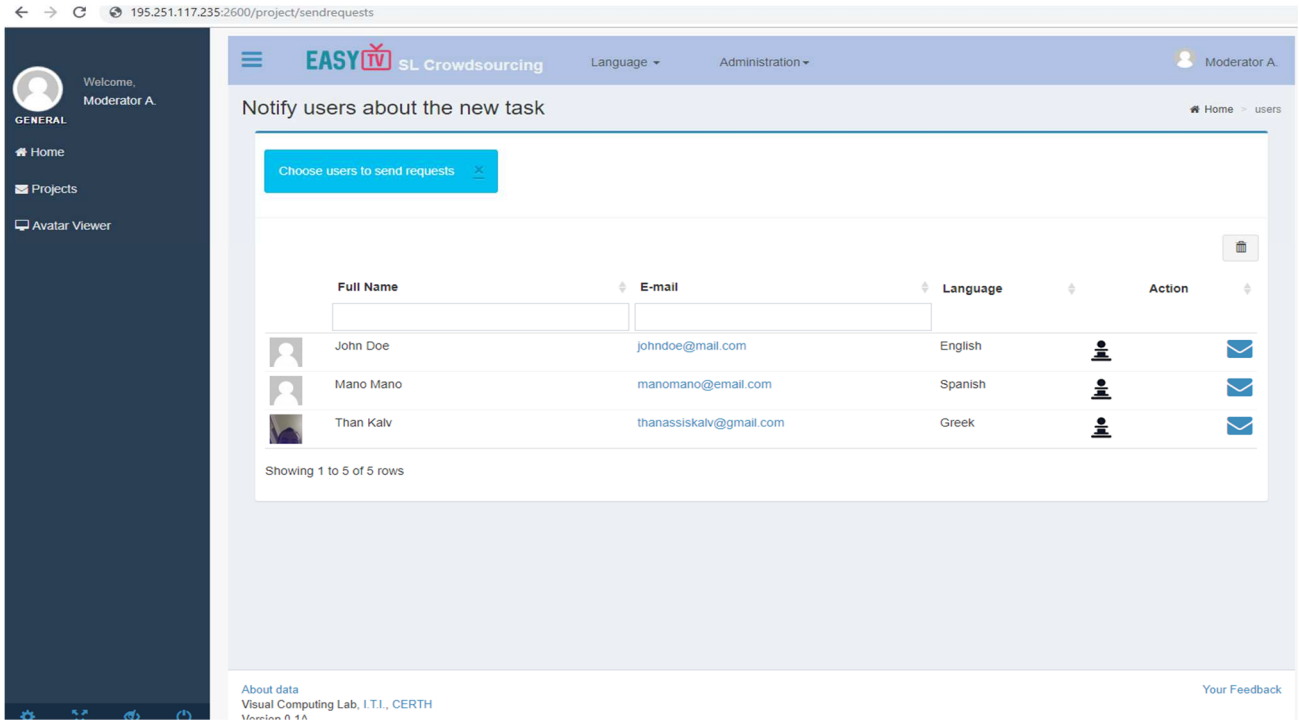


Figure 32: The moderator can notify potential users about the existence of new tasks.

Moderator reviews submissions

The third and final scenario that is presented in this deliverable involves the description of the procedures that can be followed so that a moderator can review submissions and decide to either accept or reject them. In the following steps, we describe and illustrate with screenshots from the UI elements of the EasyTV crowdsourcing platform, the procedures required from the moderator so that he/she can review user submissions.

STEP 1: The user-moderator gets notified about the existence of new material that was uploaded by a user-worker. He/she is also able to see the progress of the project in order to realize how close the project is to its completion.

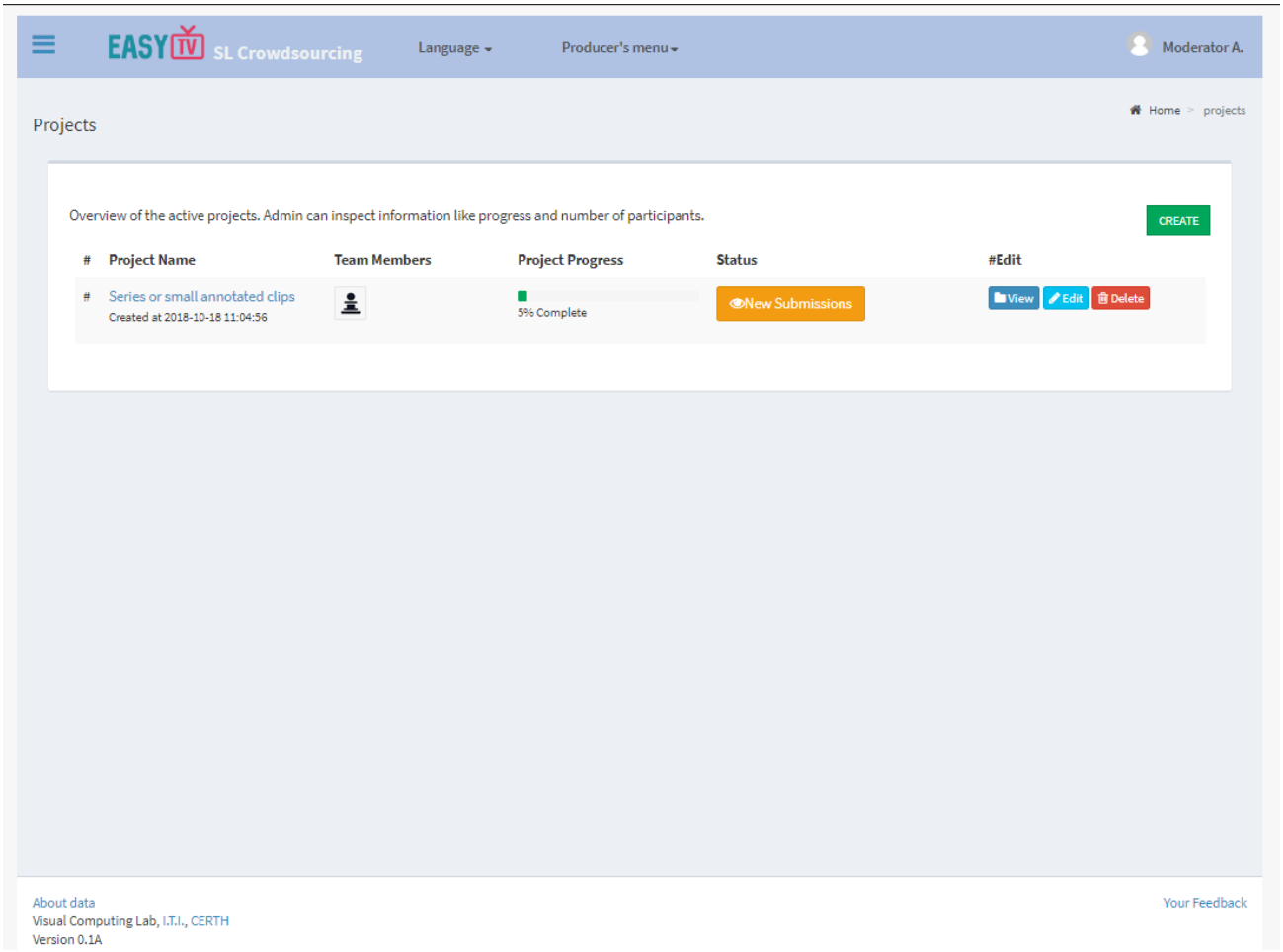


Figure 33: The moderator is notified about a new submission from a user.

STEP 2: By pressing the corresponding “View” button, a user-moderator can have access to new user submissions. In this step, the moderator can inspect the uploaded material (i.e. video, motion and annotation files) and decide whether to accept or reject the submission. The user can subsequently be informed about the decision of the moderator.

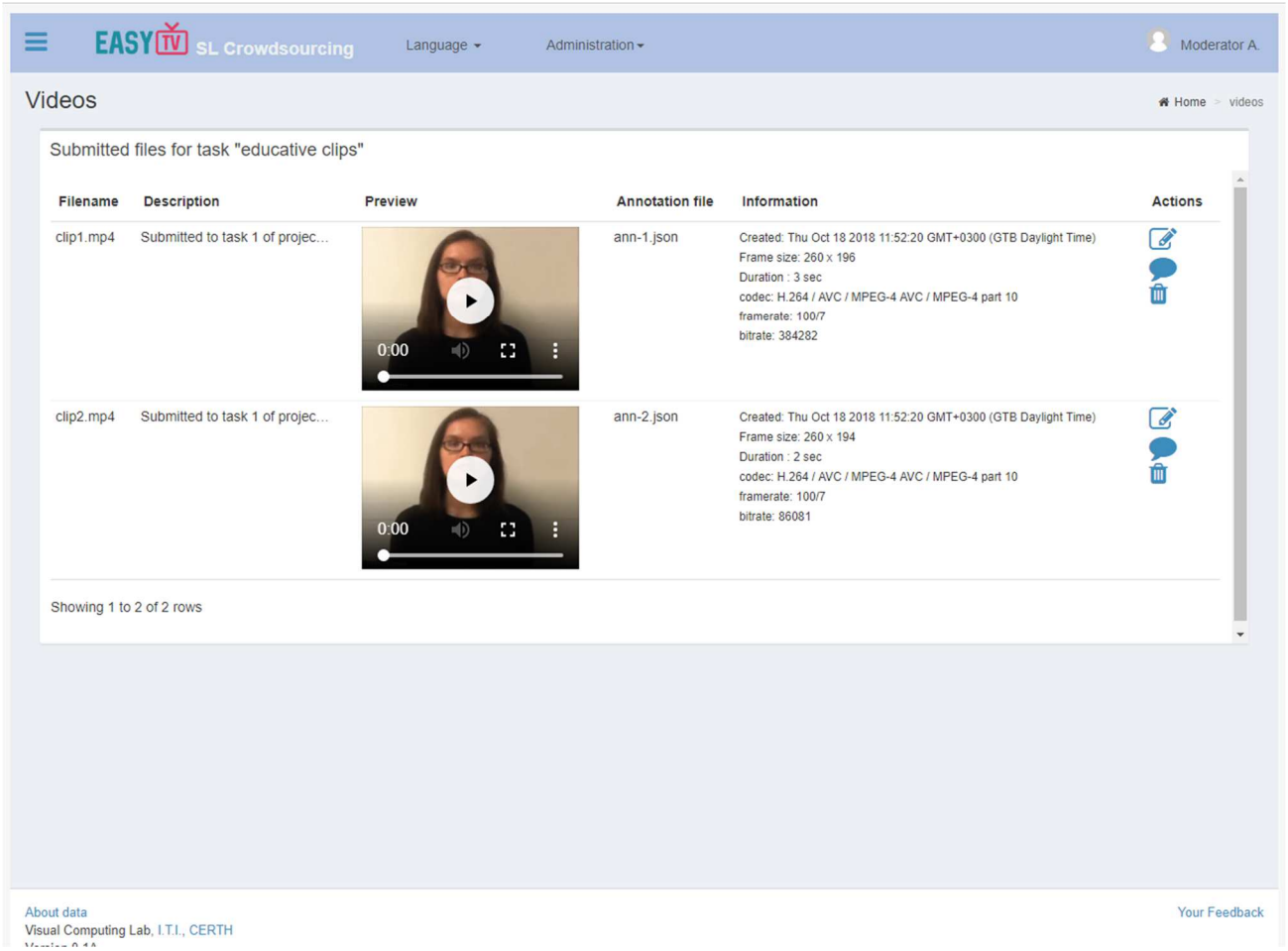


Figure 34: The moderator views the new submissions, inspects the uploaded material and decides whether to accept or reject the user submissions.