Go-Lab

Global Online Science Labs for Inquiry Learning at School

Collaborative Project in European Union's Seventh Framework Programme

Grant Agreement no. 317601

Deliverable D5.5

Release of personalisation features and inquiry learning apps - final

Editors:	Lars Bollen (UT) Adrian Holzer (EPFL)
Date:	30th June 2015
Dissemination Level:	PU
Status	Final

The Go-Lab Consortium

Beneficiary Number	Beneficiary name	Beneficiary short name	Country
1	University Twente	UT	The Netherlands
2	Ellinogermaniki Agogi Scholi Panagea Savva AE	EA	Greece
3	École Polytechnique Fédérale de Lausanne	EPFL	Switzerland
4	EUN Partnership AISBL	EUN	Belgium
5	IMC AG	IMC	Germany
6	Reseau Menon E.E.I.G.	MENON	Belgium
7	Universidad Nacional de Educación a Distancia	UNED	Spain
8	University of Leicester	ULEIC	United Kingdom
9	University of Cyprus	UCY	Cyprus
10	Universität Duisburg-Essen	UDE	Germany
11	Centre for Research and Technology Hellas	CERTH	Greece
12	Universidad de la Iglesia de Deusto	UDEUSTO	Spain
13	Fachhochschule Kärnten – Gemeinnützige Privatstiftung	CUAS	Austria
14	Tartu Ulikool	UTE	Estonia
15	European Organization for Nuclear Research	CERN	Switzerland
16	European Space Agency	ESA	France
17	University of South Wales	USW	United Kingdom
18	Institute of Accelerating Systems and Applications	IASA	Greece
19	Núcleo Interactivo de Astronomia	NUCLIO	Portugal

Contributors

Name	Institution
Lars Bollen, Jacob Sikken, Anjo Anjewierden, Siswa van Riesen, Ellen Wassink - Kamp, Henny Leemkuil, Ton de Jong	UT
Adrian Holzer, María Jesús Rodríguez-Triana, Aubry Cholleton, Alex Wild, Denis Gillet, Wissam Halimi	EPFL
Alexandros Trichos, Panagiotis Zervas	CERTH
Pablo Orduña (internal reviewer)	UDEUSTO
Georgios Mavromanolakis (internal reviewer)	EA

Legal Notices

Executive Summary

This deliverable presents the final release of the personalisation features and the inquiry learning apps developed following the specifications presented in D5.1 and the initial version released in D5.3. Personalisation is supported at many levels in Go-Lab, especially, here in this deliverable we report:

- Personalisation of **Inquiry learning Spaces** is supported in the Go-Lab Repository (Golabz) by enabling teachers to share, reuse and modify ILSs created by colleagues. Typically, the learning content can be personalised at will by customizing phases as well as the instructions and items they contain (e.g. images, files, links, apps). In particular, we detail how apps contained in ILSs can be personalised through their configurations to fit particular learning scenarios. A basic recommender system for labs and apps is also provided to support this process. This personalisation level will be presented in D5.6.
- Personalisation of apps and labs, especially at the level of **internationalisation** and terminology is supported by the App composer. This level will also be presented in D5.6.
- **Individual or collaborative personalisation of apps** (in context and content) by teachers (co-)creating an ILS for their own students is supported in the ILS Platform (Graasp) and in the apps themselves. This level is presented in this deliverable. The design of the apps themselves has been defined by WP1 and evaluated by WP8. The releases described here take into account this feedback (internally available as G8.3 and which will be finalized as D8.3).
- **Activity-based personalisation of advanced apps** relying on Learning Analytics is enabled by the LA backend. This level will be presented in D4.6 (specifications) and D4.8 (releases).

This deliverable also further details the inquiry learning apps presented in D5.3 with respect to changes, new features, and their finalisation. The inquiry learning apps have been refined with respect to their appearance and user interfaces, adapted behavior and features, and especially concerning a new, integrated functionality for adaptations and personalisation by ILS authors.

Finally, this deliverable elaborates on changes in supportive components for the inquiry learning apps that have been necessary for their adaptation. Particularly, this addresses changes in the storage mechanism, inter-app communication and the ILS library.

This deliverable does not report on apps that are related to Learning Analytics in Go-Lab, like the Reflection App or teacher dashboard apps. These will be reported in deliverable D4.8.

Table of Contents

1 Introduction

This deliverable presents the final release of personalisation features and inquiry learning apps and acts as the successor of deliverable D5.3 “Release of Personalisation Features and Inquiry Learning Apps - Initial”. It presents the developments and adaptations that followed the presentation of D5.3 and related comments from reviewers and teachers. The main product of this deliverable is the developed software. This text describes new developments and changes of the personalisation features and inquiry learning apps.

The work on personalisation and inquiry learning apps in WP5 is guided by input from WP1 and WP3. WP1 provides pedagogical guidelines to design certain components, e.g., an application to support the design of experiments. WP3 then provides feedback on the designed artefacts. For instance D3.2 details usability outcomes of all inquiry apps assessed by WP3. Since the process is iterative, WP5 discusses the outcomes with WP3 and WP1 and sheds technical challenges and opportunities on the situation. For instance, D3.2 contains evidence for this process in its Appendices E and F.

This deliverable is structured as follows: Chapter 2 focuses on the personalisation and authoring features of the Go-Lab ILS platform (i.e. Graasp). Whereas D5.3 focused on internationalisation, this deliverable takes a larger lense and presents personalisation features of ILSs focusing on the learning context (e.g., ILS name, language, background image) as well as the learning content (managing phases, items, apps and labs).

Chapter 3 provides an update on the components that support the development and the execution of inquiry learning apps. This includes changes to the inter-app communication, the new configuration and personalisation features for apps, changes and optimizations of the app resource storage facility (the Vault) and modifications and extensions to the common ILS javascript library.

Chapter 4 lists the core Go-Lab inquiry learning apps, describes their current state and the changes from the previous version as reported in D5.3 namely the Concept Mapper, the Hypothesis Tool, the Experiment Design Tool, the Data Viewer, the Drop File app, the Resource View App and the Reporting Tool.

Finally, Chapter 5 concludes this deliverable by summarizing the main aspects.

2 Personalisation and Authoring

Hereafter, we present the final release of the personalisation features in Go-Lab. In D5.3, we focused on internationalisation, in this deliverable we detail the whole authoring process of ILSs. Whether teachers reuse an existing ILS published on the Go-Lab Lab Repository, or create one from scratch, they can personalise it at will in Graasp, the ILS Platform. For instance, they can modify the *context* (e.g., language, title, background), as well as the pedagogical *content* (e.g., resources, descriptions, phases). As detailed in D5.4, teachers can publish their personalized ILS back on the Lab Repository.

Personalising context

ILSs can be renamed by clicking the title. Their background image can be replaced in the editing tab of the ILS. The left side of Figure 1 shows how to change the title. The right side shows the place where the background image can be selected (a different one can be chosen for each phase). Note that ILSs can be created collaboratively by several teachers. The author of an ILS can simply add others as members to the ILS with appropriate permissions for that purpose.

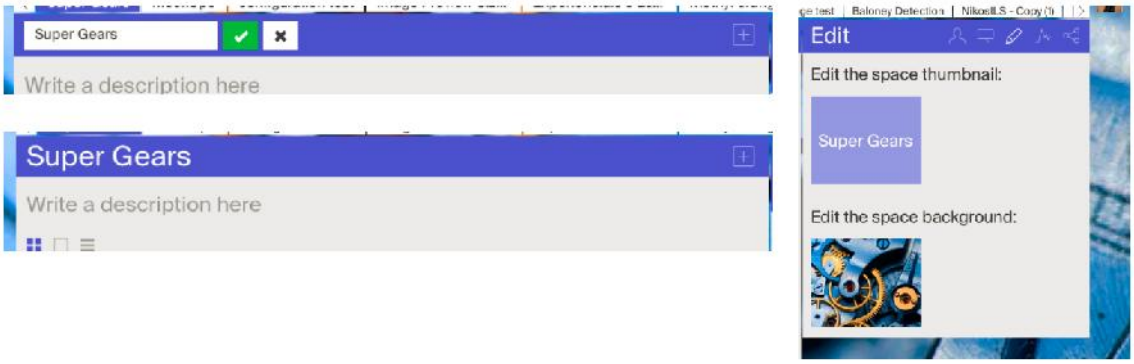


Figure 1. Changing title and background image.

Figure 2 shows an example of a custom background image in the Standalone View of the Super Gears ILS.

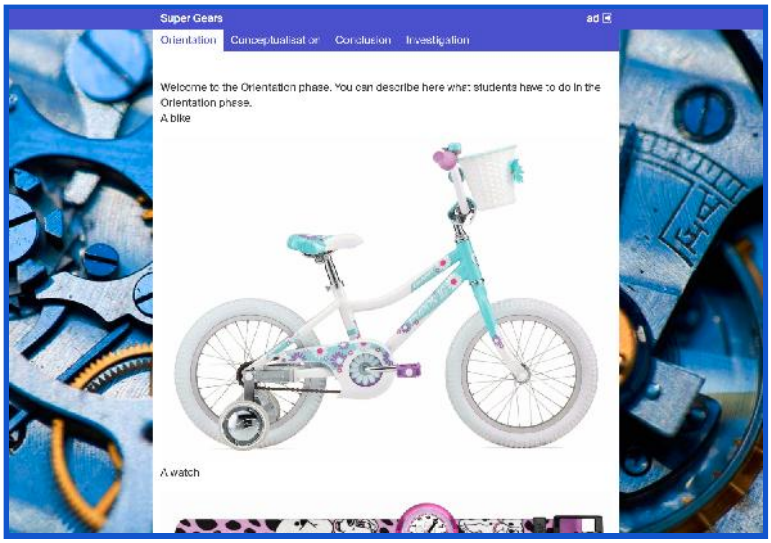


Figure 2. Background image of an ILS seen in the Standalone View.

The **language** of the ILS Standalone view can be set when sharing the ILS with students by selecting the adequate language in the Sharing tab. Changing the language of the ILS means that the User Interface of the Standalone view will be translated as well as all inquiry learning apps inside (given that a translation in the particular language exists). Translations for apps are provided via the App Composer as documented in D5.4. As mentioned in D5.3 the content of ILSs will be translated by the teacher. The left side of Figure 3 shows where the language can be selected and the right side shows the adapted Standalone View welcome screen in french.

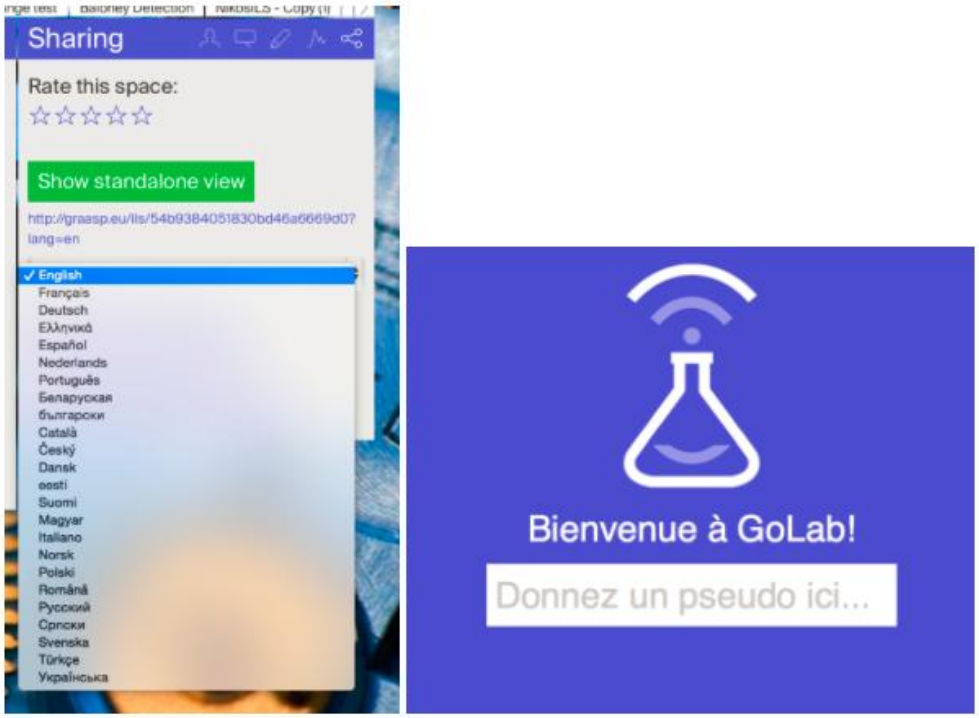


Figure 3. Selecting the language of the ILS.

Personalising content

The pedagogical content of an ILS can also be extensively personalised. Teachers can manage phases, their items, and description. Teachers can manage **phases** by reordering existing phases by drag and dropping them as shown in Figure 4 below on the left, removing them by pressing the (so-called hamburger) menu on the top right of an item (see the figure below on the right).

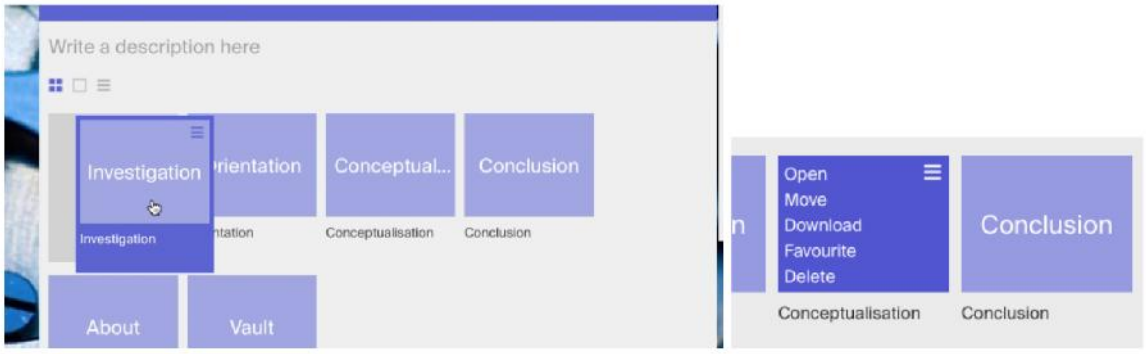


Figure 4. Moving phases of the ILS

As illustrated in Figure 5, teacher can also add new phases by pressing the plus button on the top right of the space and press the create space button (a phase is instantiated through a space in the ILS Platform).

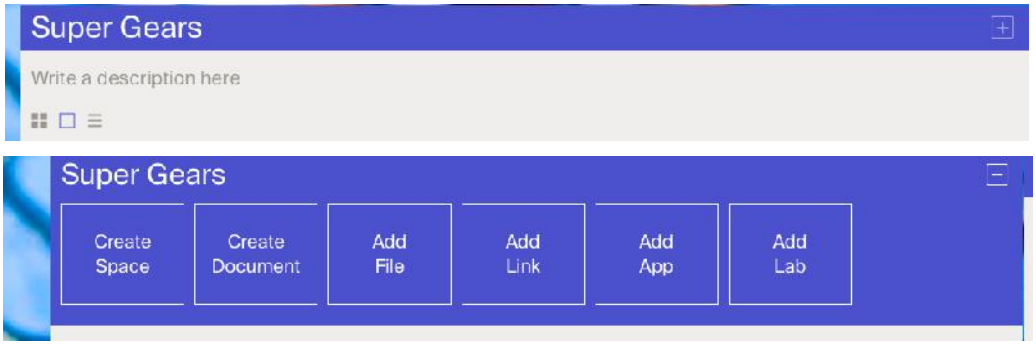


Figure 5. Adding content to a space.

In order to add **items** in a phase, teachers can either drag and drop files (e.g., pdf, or images) into any phase of the ILS or press the plus button on the top right of the space content and press on the relevant button (create document, add file, add link, add app, or add labs).

In order to create a new file, the *create document* button can be pressed (in order to add an existing file, the *add file* button can be pressed). When pressing create document, a user can choose the extension of the new document to be created (e.g., graasp, html, txt). Graasp documents are rich text files that allow to add tables and basic text formatting as shown on Figure 6.

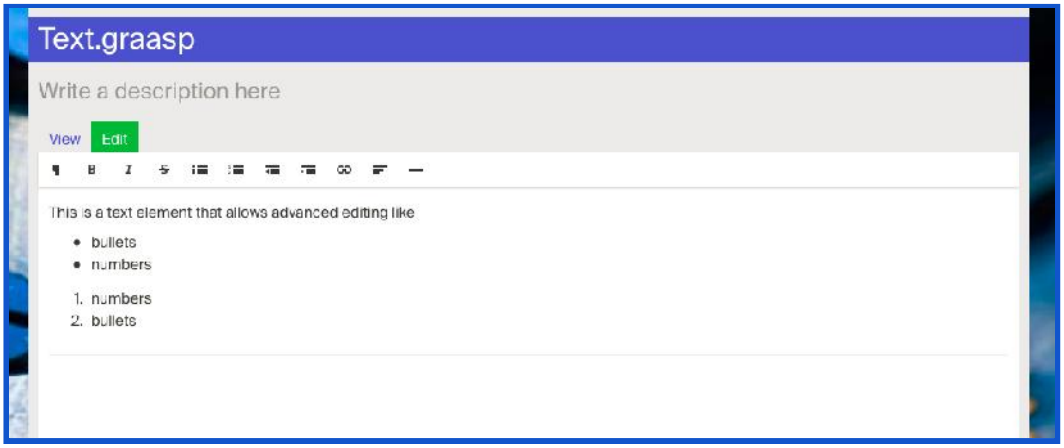


Figure 6. An example of a GRAASP document.

With HTML documents, the edit tab of the document illustrated in Figure 7 allows to write code and the view tab provides a view of a rendering of the code as shown on Figure 8.

Figure 7. Edit mode of an HTML document.

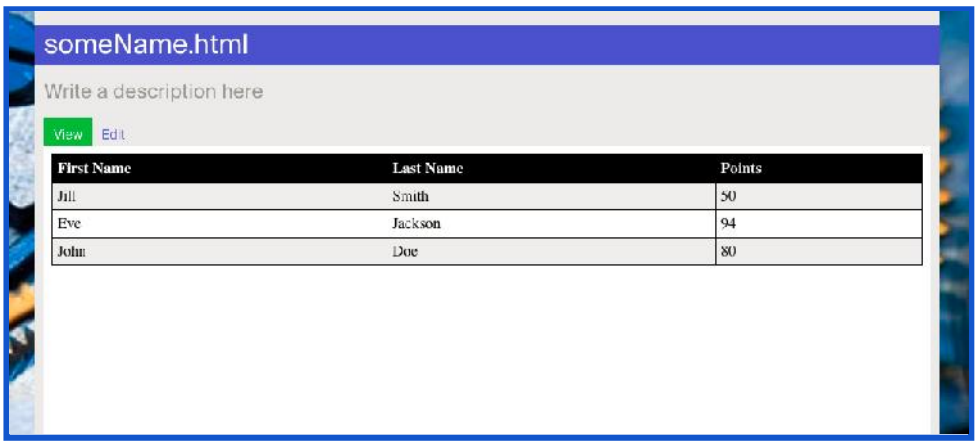


Figure 8. View mode of an HTML document.

Links to web pages can also be added in an ILS by pressing the *add link* button. Pressing this button will embed the target page into the phase. For example Figure 9 shows an embedded Youtube video.

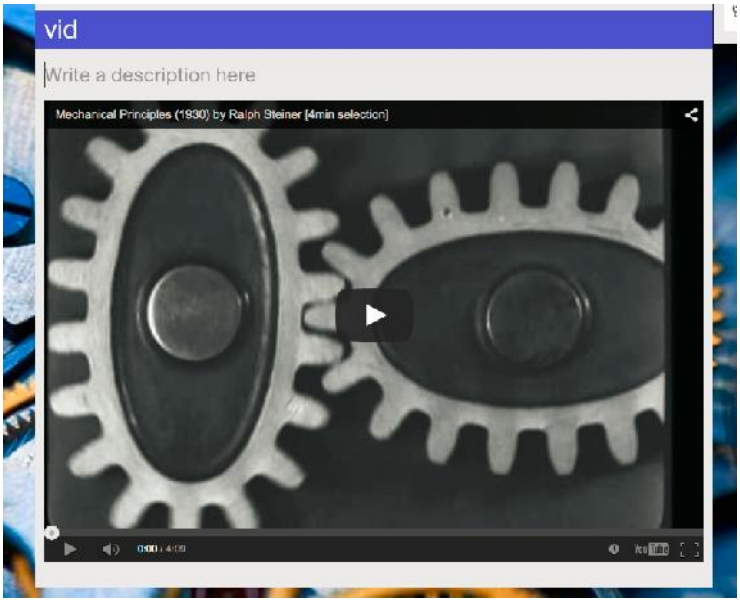


Figure 9. Example of an embedded Youtube video.

Apps and Labs can also be added to any phase by pressing the *add lab* or *add app* button. After pressing the button, teaches can either provide a link to a lab or an app of their choosing, or simply select the app or lab in the list of labs and apps available on the Lab Repository (golabz.eu) as shown on Figure 10.

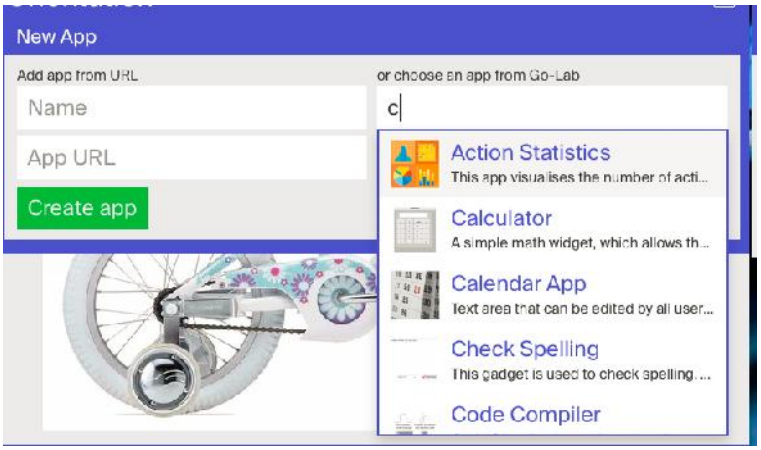


Figure 10. List of apps available displayed when adding a new app.

Files such as images can be resized in three standard formats (small, medium, large) by pressing the pencil icon on the top right of the item bar in the expanded view as shown on Figure 11.

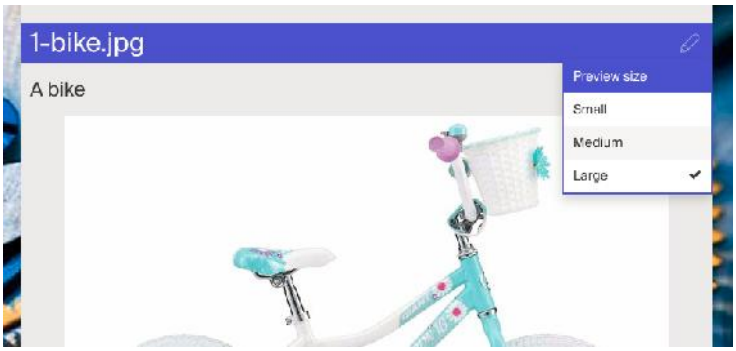


Figure 11. Resizing options for images.

Each item (i.e., file) has a description that can be modified by simply pressing on it and style can be applied with short cuts bold (ctrl+B) italic (ctrl+I) and hypertext links (ctrl+K) as illustrated in Figure 12. Furthermore, the description supports the creation of inline and centered equations written in Latex. A simple inline equation uses the `\(` opening tag and the `\)` closing tag. Centered equation use the `\[` opening tag and the `\]` closing tag.

Figure 12. An example of a description of an item.

3 Supporting Inquiry Learning Apps

This chapter describes supportive components which are needed at various stages of an app's lifecycle, e.g. during authoring, in the Standalone View, or while storing and retrieving resources. Particularly, the changes and updates made with respect to previous descriptions from D5.3 are listed here.

Inter-app Communication Library

The communication between inquiry learning apps, as described in D5.3, section 3.1, has been replaced by the sole use of the Go-Lab Learning Analytics framework. In brief, all inquiry learning apps are supporting user activity logging (as described in D5.3, section 3.3), and those apps who are interested in receiving information make use of the Notification infrastructure. This architecture has been described in D4.2, section 2.4 and will be further elaborated on in D4.4 and D4.6.In particular, the Reflection apps (which will be described in D4.4) are making use of this infrastructure by processing and visualizing information from various inquiry learning apps for the purpose of creating Learning Analytics dashboards for teachers and learners alike.

App Personalisation

One important, major new personalisation feature presented in this deliverable is the configuration option, which allows to personalise the inquiry learning apps by enabling or disabling features, and by adapting the apps towards targeted learning domains, e.g., by choosing the shown domain terms in the Hypothesis Scratchpad or by defining available variables in the Experiment Design Tool. Similarly, a customized help text can be created to suit the needs of specific ILSs and learning domains.

The data for app personalisations is automatically attached and stored directly to an app - in contrast to the Vault storage (see below), which is used to store students' artefacts while working with the apps. Thus, an ILS can be easily copied and duplicated without copying private student data, while keeping the apps' configurations available in the copied ILS as well.

In a nutshell, with the personalisation features of apps, we have integrated the adaptor feature of the AppComposer directly into the ILS Platform.

The following two screenshots (Figures 13 and 14) demonstrate how an app (in this case, the Hypothesis Scratchpad) can be configured during authoring in the ILS Platform, and how the result looks in the student view. In the example, the Hypothesis Scratchpad is configured to be used in an ILS in the domain of "radioactivity", where terms like "exposure time" or "absorber's atomic number" are of relevance. Additionally, an unfinished hypothesis has been prepared, which will be shown at startup to the students.

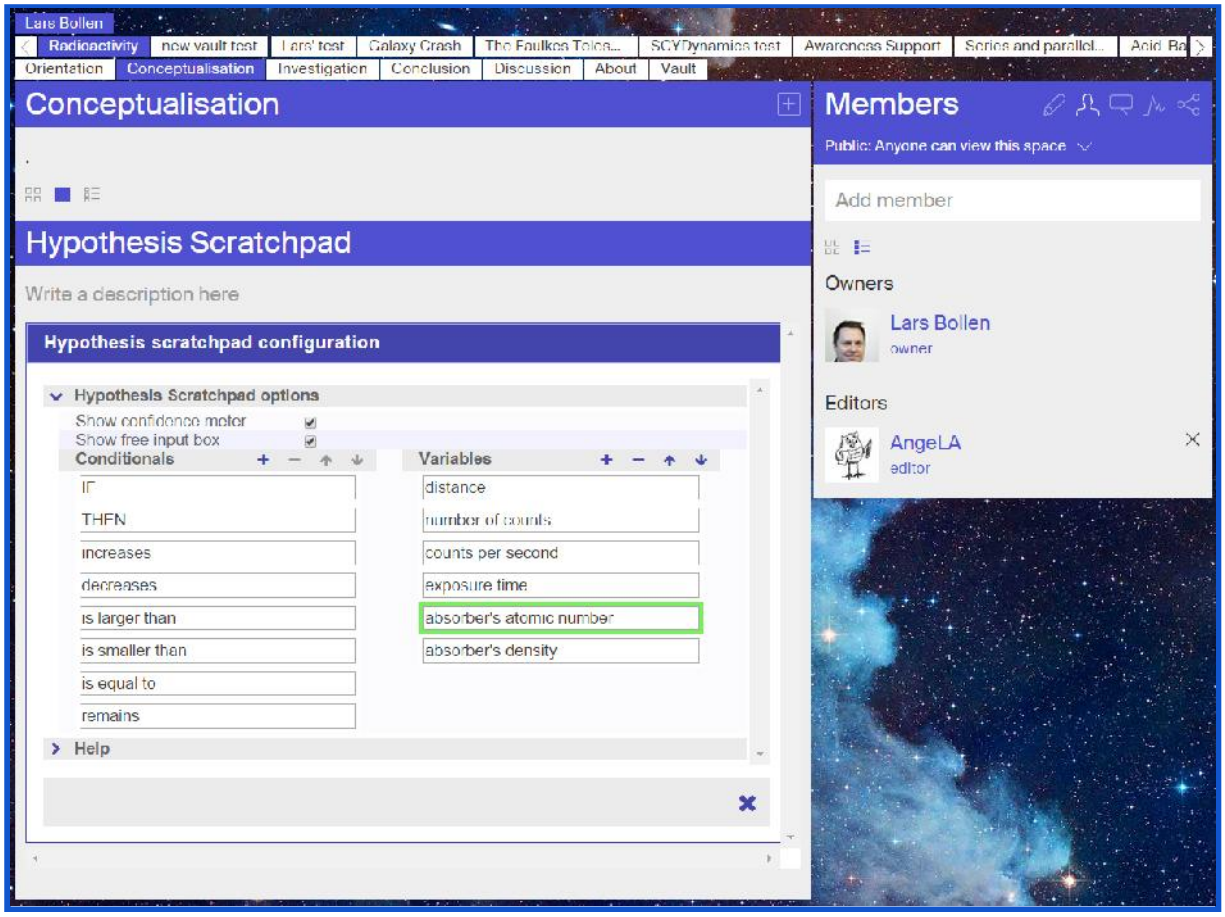


Figure 13. Configuring an inquiry learning app during authoring.

Figure 14. The resulting configured Hypothesis Scratchpad in the student view.

App Storage: The Vault

As it was already presented in D5.3, the Vault space is envisioned to allow apps to store data. This service supports different needs such as the possibility to provide pre-made resources to the students, saving the student work, and allowing the data exchange between apps, while preserving the privacy policies defined in the project.

To support app developers in the use of the Vault space, the ILS library and the StorageHandler have been extended to provide services for the creation, update, and retrieval of resources (see below). Typically, an app developer makes use of the StorageHandler API as described in D5.3 (accessible online through the following link https://github.com/go-lab/ils/wiki/StorageHandler), whereas the StorageHandler forwards and delegates function calls and results. Thus, different implementations of the storage mechanism can be implemented transparently to the apps.

Caching mechanism

As a response to teachers' comments about occasionally slow loading times of apps and ILSs as a whole, a caching mechanism has been implemented for the apps' access to the Vault. During the startup of an ILS, a learner's resources are once downloaded into the browser's local storage facility. Every subsequent access to the Vault will then be handled through the much faster, local browser storage. Write access (i.e. create or update a resource) will change the local storage and the (remote) Vault storage, and updates of the same resource in short succession are bundled into one remote call.

ILS library

URL of the source code repository: https://github.com/go-lab/ils
Library documentation: https://github.com/go-lab/ils/wiki/ILS-Library

As defined in D5.3, the ILS library has been developed in JavaScript and uses the OpenSocial API to communicate the apps with Graasp. From a practical point of view, the purpose of the ILS library is to support the data sharing between the apps and Graasp, namely, the resources generated by the students, the configurations of the apps, the actions carried out by the users and the information about the context where the apps are embedded.

From a more technical perspective, the following tables show the whole list of implemented methods. The new ones appear highlighted in bold. In terms of **data storage**, the ILS library supports both the management of resources generated by the apps and the storage of app configurations (see Table 1). To support the resource lifecycle, the new methods allow not only creating and accessing existing resources but also updating and removing them as well as accessing their metadata. Apart from that, new methods have been developed in order to allow the apps to store and retrieve their configurations or even have access to the configurations of other apps located in the same ILS. It is noteworthy that while the

configurations are stored internally in the Graasp database as app attributes, the resources are represented as files in the Vault. This schema allows us to copy or move the apps to different spaces -or even to different ILSs- while preserving their configurations. Regarding the resources stored in the Vault, in order to ensure their privacy, they are never moved or published. That means that if the app is moved to another ILS, the app will not be able to access the existing resources of the previous ILS. In case of publishing an ILS, the configuration of the apps will be maintained but the new Vault space will be empty, since its original content will not be published.

Table 1. ILS library methods for storing and retrieving resources and app configurations

Data Storage Methods	
createResource	Creates a new resource in the Vault
existResource	Verifies the existence of a resource based on its identifier
readResource	Returns a resource in the Vault based on its identifier
getMetadata	Returns the metadata of a resource based on its identifier
updateResource	Updates an existing resource in the Vault
deleteResource	Removes an existing resource in the Vault
listVault	Returns the list of resources stored in the Vault
getConfiguration	Returns the app configuration
setConfiguration	Sets the app configuration
listAllConfigurations	Returns the configurations of all the existing apps in the ILS

For a number of features that support personalisation or configuration, the apps need to be aware of the **context** where they are running, e.g. to differentiate between the Authoring Mode and Standalone View. The methods listed in Table 2 have been implemented to inform the apps about the context where they are rendered and to collect the information they may need to adapt their behavior, manage the resources or log the user actions. For example, the *identifyContext* method allows the apps show different information to teachers or students depending on where they are being executed. On the other hand, the *getAppContextParameters* method provides the necessary information to use the *ActionLogger* and the *MetadataHandler* libraries.

Thanks to these functionalities, apps such as the Concept Mapper or the Hypothesis Scratchpad provide additional features e.g. configuration menus when they are rendered in the Authoring Mode (see Section 4 for further details).

Table 2. ILS library methods to identify the execution context and to retrieve context information

Context Methods	
getCurrentUser	Returns the nickname of the current user
identifyContext	Returns the context where the app is being rendered, namely Golal ("preview"), Graasp ("graasp" or "standalone_ils"), a non open social conte ("standalone_html"), or none of the previous ones ("unknown")
getParent	Returns the metadata of the space where the app is located
getParentInquiryPhase	Returns the type of the phase where the app is located, namely: "Orientatior "Conceptualisation", "Investigation", "Conclusion", "Discussion", "About "Vault", or “undefined”, if it is a custom phase
getIls	Returns the metadata of the ILS where the app is located
getSpaceBySpaceId	Returns the metadata of a space based on its identifier
getItemsBySpaceId	Returns the list of items in a space based on its identifier
getSubspacesBySpaceId	Returns the list of subspaces in a space based on its identifier
getAppsBySpaceId	Returns the list of apps in a space based on its identifier
listFilesBySpaceId	Returns all the resources stored in a space based on its identifier
getVault	Returns the metadata of the Vault space assign to the app to store resources
getVaultByIlsId	Returns the metadata of the Vault based on its identifier
getApp	Returns the metadata of the current app
getAppId	Returns the metadata of an app based on its identifier
getAppContextParameters	Returns the description of the app context, namely "actor", "generator "provider", "target", "storageId" and "storageType"
setContextParameters	Sets the attributes of the app context.

Dealing with personalisation, one of the objectives for this release has been to allow students to store and retrieve their own resources each time the open an ILS, and, at the same time, allow teachers to visualise the resources generated by the students in the different apps. From a technical point of view, it implies that the apps should be able to filter their own resources from the Vault. This has been realized by attaching detailed metadata to the resources when they are created and updated. Combining the methods for context awareness with the analysis of the resource metadata, the apps can either show to the students their own resources, or collect and display all produced artefacts to the teacher. The usage of these functionalities is illustrated in the Input Box description offered in the following section.

4 Inquiry Learning Apps

This chapter describes the final state of the inquiry learning apps, which includes an adjusted design, and new or modified features after taking into account feedback from teachers and ILS authors.

In the following, the Go-Lab core inquiry learning apps will be listed in their final release version. The descriptions from previous deliverables, in particular those from D5.3, will not be repeated if still valid. Here, only relevant changes and updates will be described, most of which are direct results from the participatory design activities and recommendations reported in D3.2. Please note that the Wiki App, which has been described in D5.3, has not been changed significantly and thus will not be listed in this deliverable. The supported languages only denote those which are available by default as a requirement from past studies and experiments; more languages can be easily added through the AppComposer Translator, which will be fully described in D5.6.

Concept Mapper

URL on the lab repository: http://www.golabz.eu/content/go-lab-concept-mapper
Supported languages: English, German, Spanish, Estonian, Dutch
Predominant phase(s): Orientation, Conceptualisation
Screenshot: Figures 15 and 16

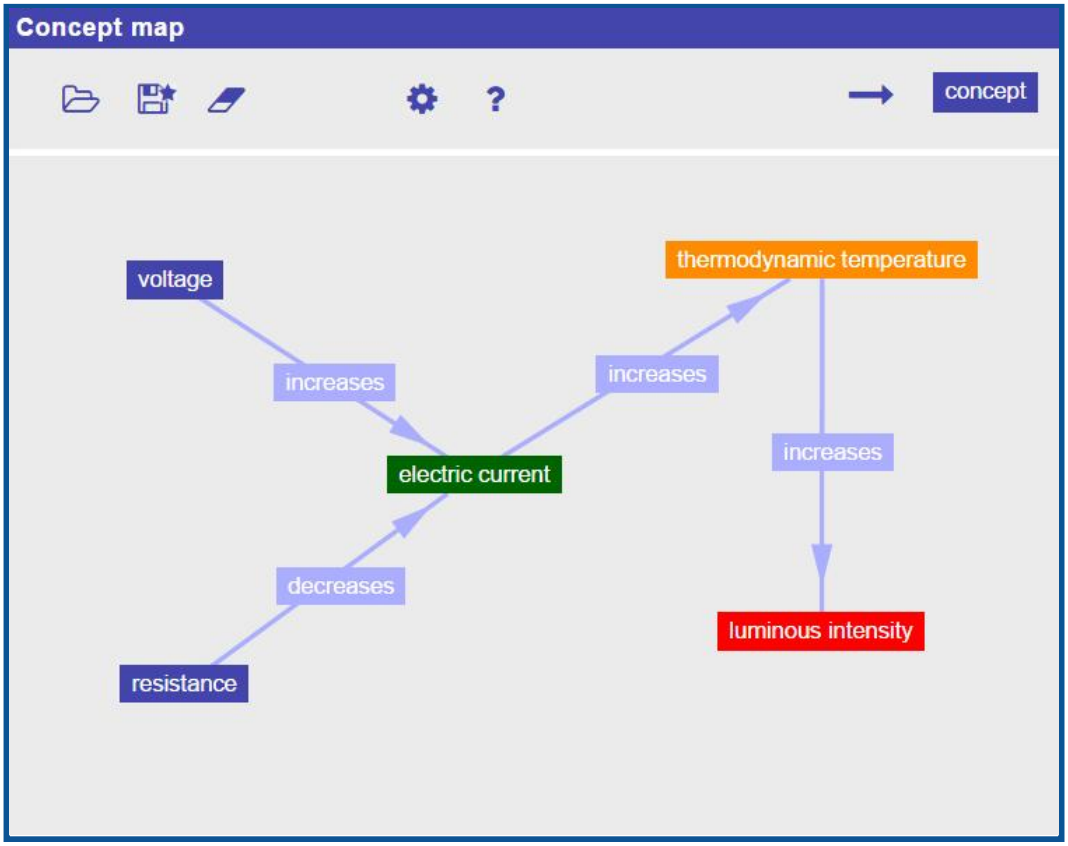


Figure 15: Screenshot of the final release of the Concept Mapper

Changes from previous versions:

- The design and color scheme has been modified to match the new Graasp user interface.
- Responsive design compliant with the Graasp style.
- Since results from usability studies indicated that deleting concepts and relations was not intuitive enough, they can now be deleted by simply dragging them back to the toolbar.
- The Concept Mapper is making full use of the Vault storage implementation, which includes an auto-save feature (i.e. every change to the concept map will be automatically propagated and stored) and an auto-load feature (i.e., the latest concept map will be re-opened on startup).
- The previous approach of configuring the Concept Mapper's properties through the AppComposer has been redesigned and directly integrated into the tool (see Figure 14). During authoring, the sets of pre-defined labels for concepts and relations can configured and tailored towards the ILS domain, along with a customized help text.
- Additionally, a pre-defined concept map can be specified during authoring, which will be loaded as the initial concept map for each learner.

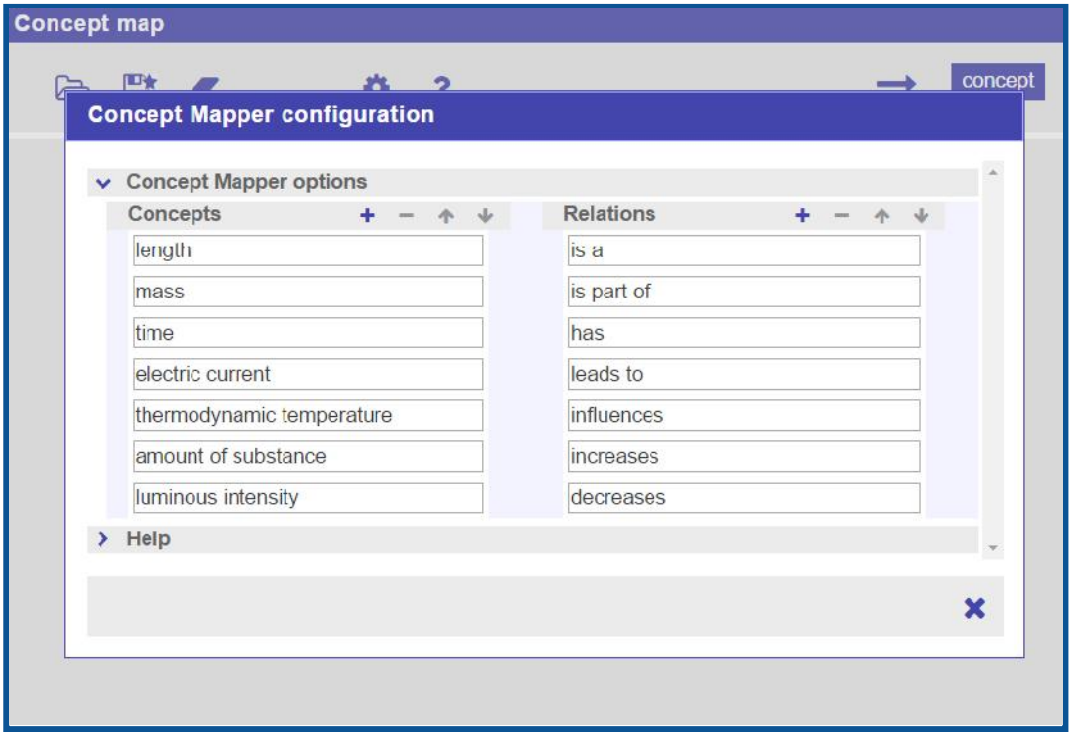


Figure 16: Configuration dialogue of the Concept Mapper tool

Hypothesis Scratchpad

URL on the lab repository: http://www.golabz.eu/app/hypothesis-tool
Supported languages: English, German, Spanish, Estonian, Dutch, Greek
Predominant phase(s): Orientation, Conceptualisation
Screenshot: Figures 17 and 18

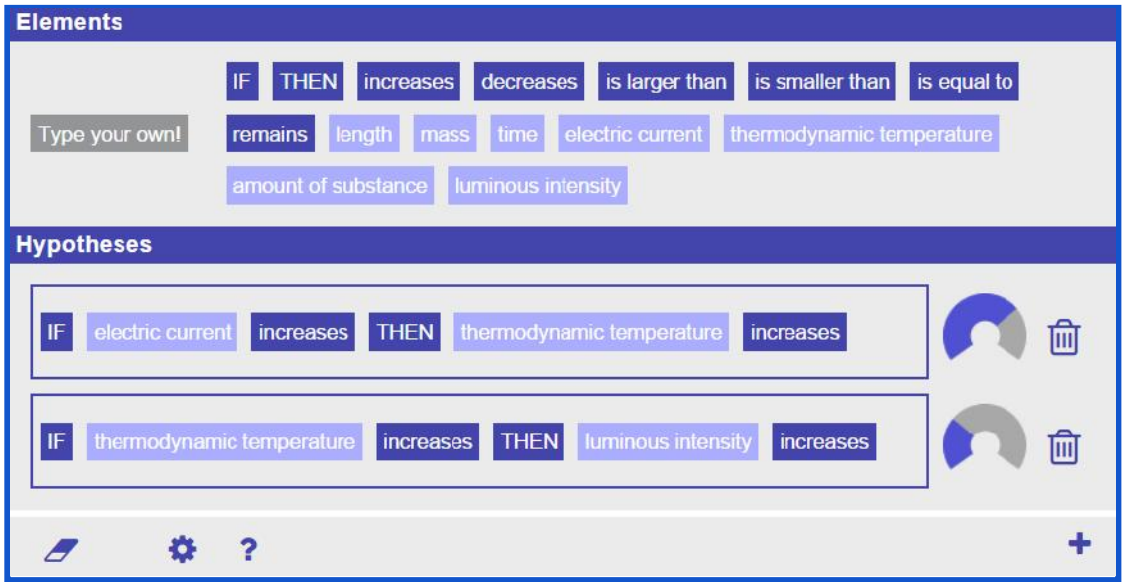


Figure 17. Screenshot of the final release of the Hypothesis Scratchpad

Changes from previous versions:

- The design and color scheme has been modified to match the new Graasp user interface.
- Responsive design compliant with the Graasp style.
- Next to each hypothesis, a "confidence meter" has been added as a way to let learners explicitly indicate their confidence in a given hypothesis. In the Conclusion Tool, this information will be picked up again to foster a learner's reflection activities.
- The Hypothesis Scratchpad provides an integrated configuration dialog, which enables teachers to customize the available domain terms and help texts. As a request from teachers, the confidence meter and the custom "Type your own!" input box can be disabled in the configuration dialog (see Figure 16).
- The Hypothesis Scratchpad is making full use of the Vault storage implementation, which includes an auto-save feature (i.e. every change to the hypotheses will be automatically propagated and stored) and an auto-load feature (i.e. the latest set of hypotheses will be re-opened on startup).

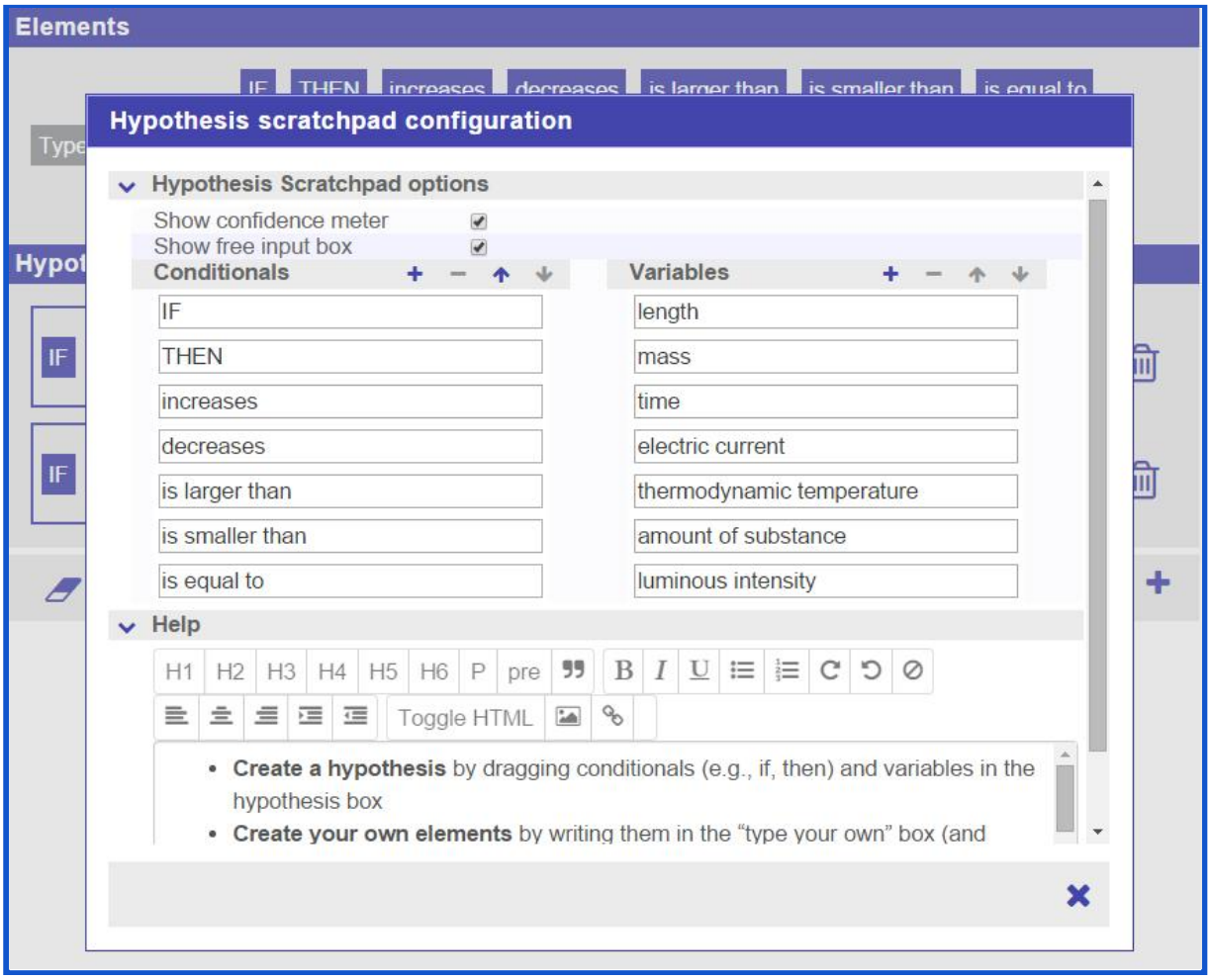


Figure 18. Configuration dialogue of the Hypothesis Scratchpad

Question Scratchpad

URL on the lab repository: http://www.golabz.eu/apps/question-scratchpad
Supported languages: English, German, Spanish, Estonian, Dutch
Predominant phase(s): Orientation, Conceptualisation
Screenshot: Figure 19

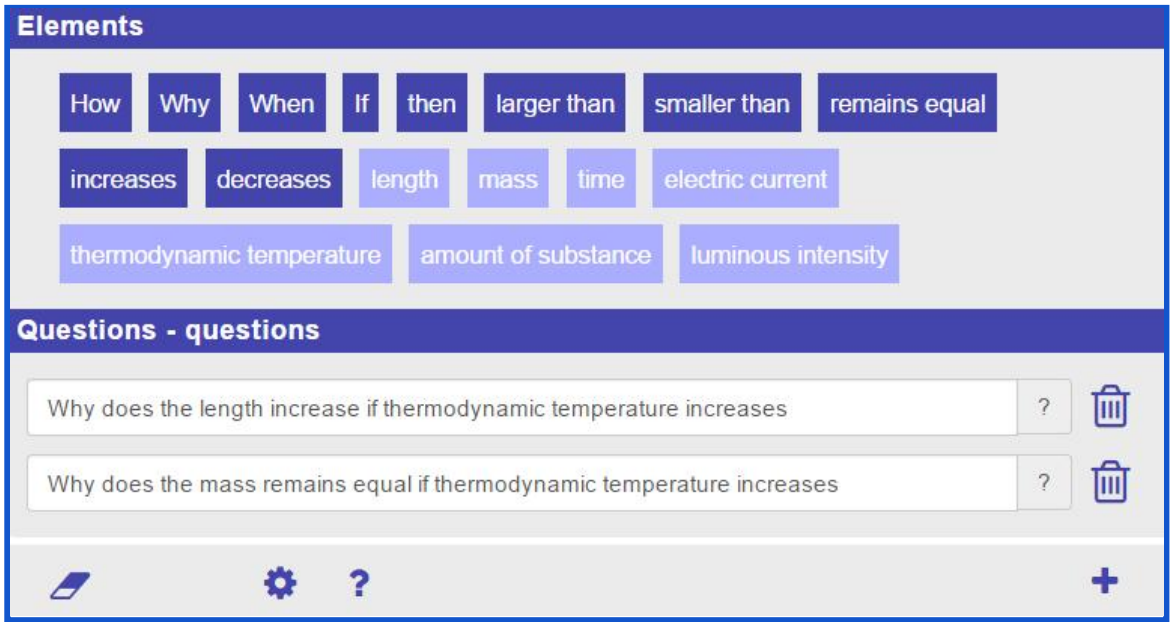


Figure 19. Screenshot of the final release of the Question Scratchpad

Changes from previous versions:

- The design and color scheme has been modified to match the new Graasp user interface.
- Responsive design compliant with the Graasp style.
- The Question Scratchpad provides an integrated configuration dialog, which lets an ILS author customize the available domain terms and help texts, similar to the ones shown in Figure 16 and Figure 17.
- The Question Scratchpad is making full use of the Vault storage implementation, which includes an auto-save feature (i.e. every change to the questions will be automatically propagated and stored) and an auto-load feature (i.e. the latest set of questions will be re-opened on startup).

Experiment Design Tool (EDT)

URL on the lab repository: http://www.golabz.eu/apps/experiment-design-tool
Supported languages: English, Spanish, Greek, Estonian, Dutch
Predominant phase(s): Conceptualisation, Investigation
Screenshot: Figure 20 and 21

The Experiment Design Tool has been completely redesigned based on the experiences with students and other feedback, making it simpler and more straightforward to use (see Figure 20). One major additional feature is in integrated configuration and personalisation function, so that teachers can completely shape it to the domain at hand (see Figure 21). As a result, the EDT is now a generic scaffolding app that can be personalized to a group of students and a domain. In addition, the teacher can now also provide students with a specified experimental set-up as a start that students can use and build upon in a progressively autonomous way.

Being a central scaffolding app for the planning of experiments, the EDT has been extensively used and evaluated in different studies by different consortium partners with altogether more than 500 students, so it can be stated that the EDT is widely experimented. The general, brief conclusion from these studies is that the EDT is most effective for students with poor prior domain knowledge or from lower levels of education. More details on the conducted studies can be found in the upcoming deliverable D8.3.

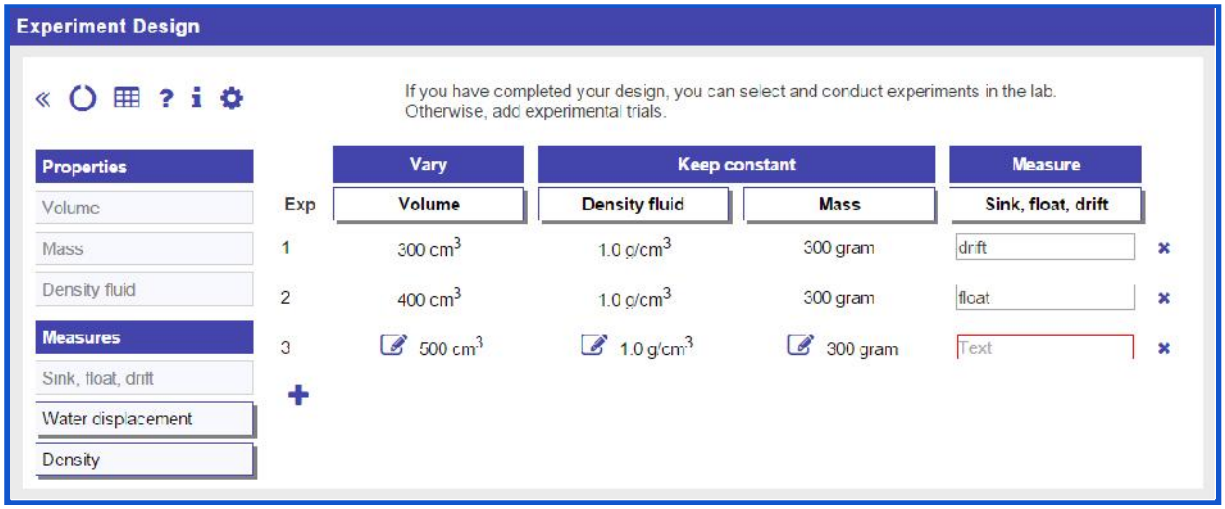


Figure 20. Screenshot of the Experiment Design Tool

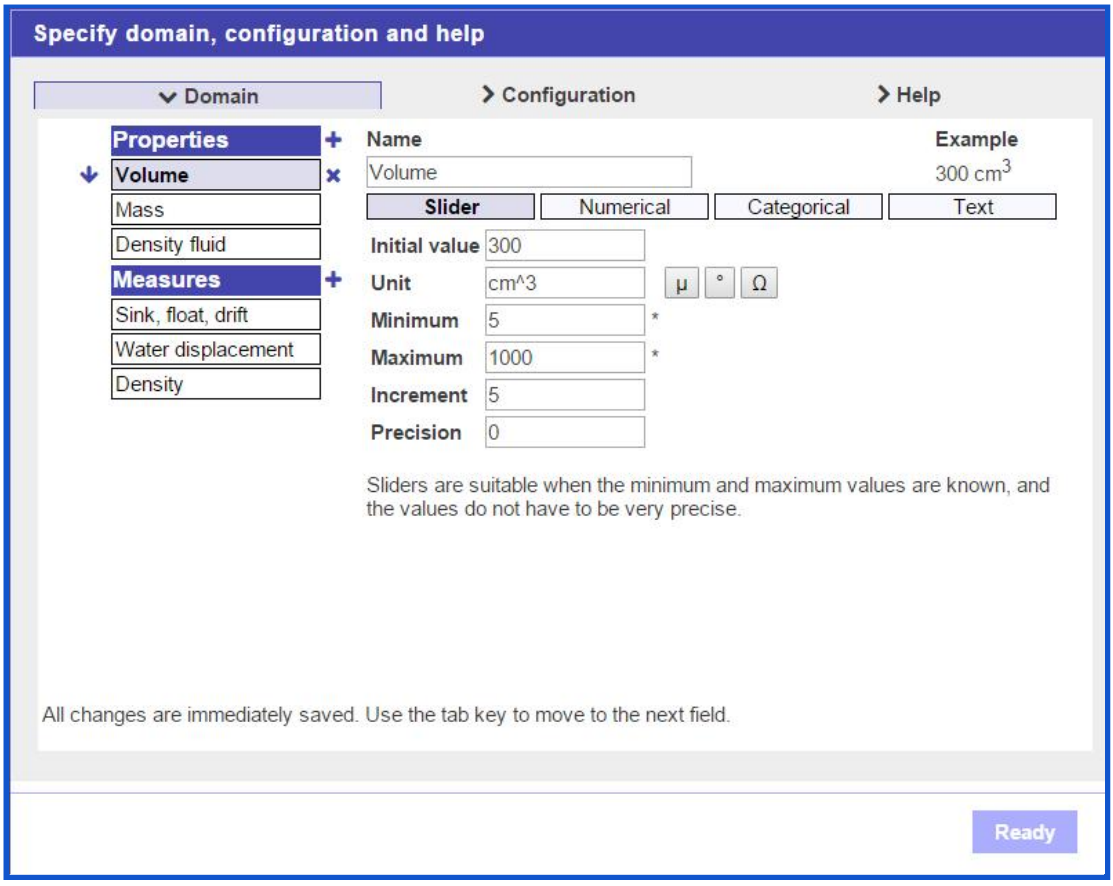


Figure 21. Screenshot of the personalisation dialog of the EDT

Changes from previous versions:

- The design and color scheme has been modified to match the new Graasp user interface.
- Responsive design compliant with the Graasp style.
- The EDT provides an integrated configuration dialog, which enables a teacher to customize the available variables (properties and measures) and help texts. Additionally, the pre-defined experiment design can be specified for the learners.
- The EDT is making full use of the Vault storage implementation, which includes an auto-save feature propagated and stored and an auto-load feature. The EDT operates with two different types of resources, namely *experiment designs* (the specification of an experiment) and *datasets* (the resulting variable measures from a conducted experiment). Datasets can be processed and visualized in the Data Viewer (see below).

Conclusion Tool

URL on the lab repository: http://www.golabz.eu/apps/conclusion-tool
Supported languages: English, Spanish, Estonian, Dutch, Portuguese
Predominant phase(s): Conclusion
Screenshot: Figure 22

From being reported as a mock-up in the previous deliverable D5.3, the Conclusion Tool has matured to its final state as shown in Figure 22. Within the Conclusion Tool, the learner can collect, visualize and reflect upon resources from other phases. The Conclusion Tool automatically includes the latest version of a learner's hypotheses (from the Hypothesis Scratchpad) and research questions (from the Question Scratchpad).
To support the learner's argumentation and conclusions, he/she can load data visualizations (from the Data Viewer) and observations (from the Observation Tool) into the Conclusion Tool. If present, the learner's confidence in his/her original hypothesis can be adjusted and justified, thus triggering reflection on the hypothesis creation and experimentation process.

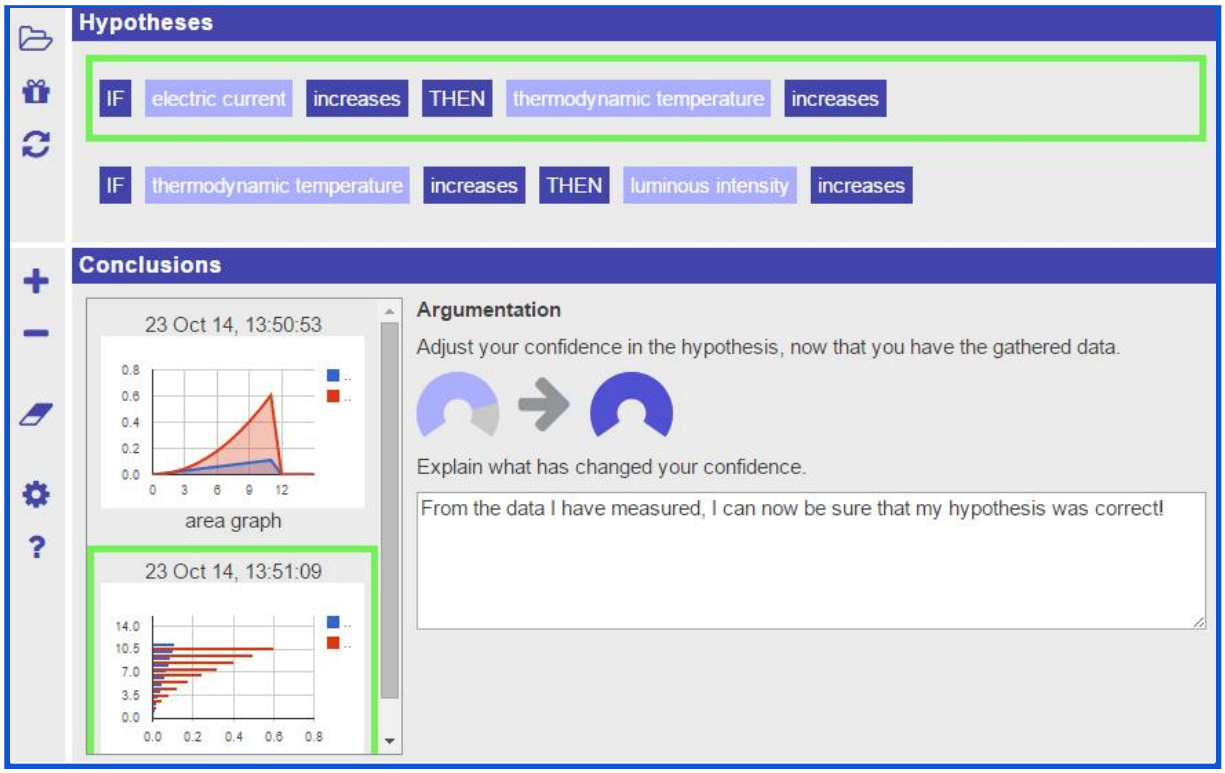


Figure 22. Screenshot of the Conclusion Tool

The main features of the newly implemented Conclusion Tool are:

- The Conclusion Tool provides an integrated configuration dialog, which enables a to teacher customize the available features, e.g. to enable/disable the inclusion of data graphs, observations or the confidence in given hypotheses.
- The Conclusion Tool is making full use of the Vault storage implementation, which includes an auto-save feature and an auto-load feature.

Data Viewer

URL on the lab repository: http://www.golabz.eu/apps/data-viewer
Supported languages: English, Spanish, Estonian, Dutch
Predominant phase(s): Investigation, Conclusion
Screenshot: Figure 23 and 24

Apart from changes in appearance and user interface (see Figure 23), the Data Viewer is now capable of editing datasets as well (see Figure 24). This enables learners to change or add data points, e.g. from experiments that have been conducted in remote or physical labs which do not automatically create data sets.

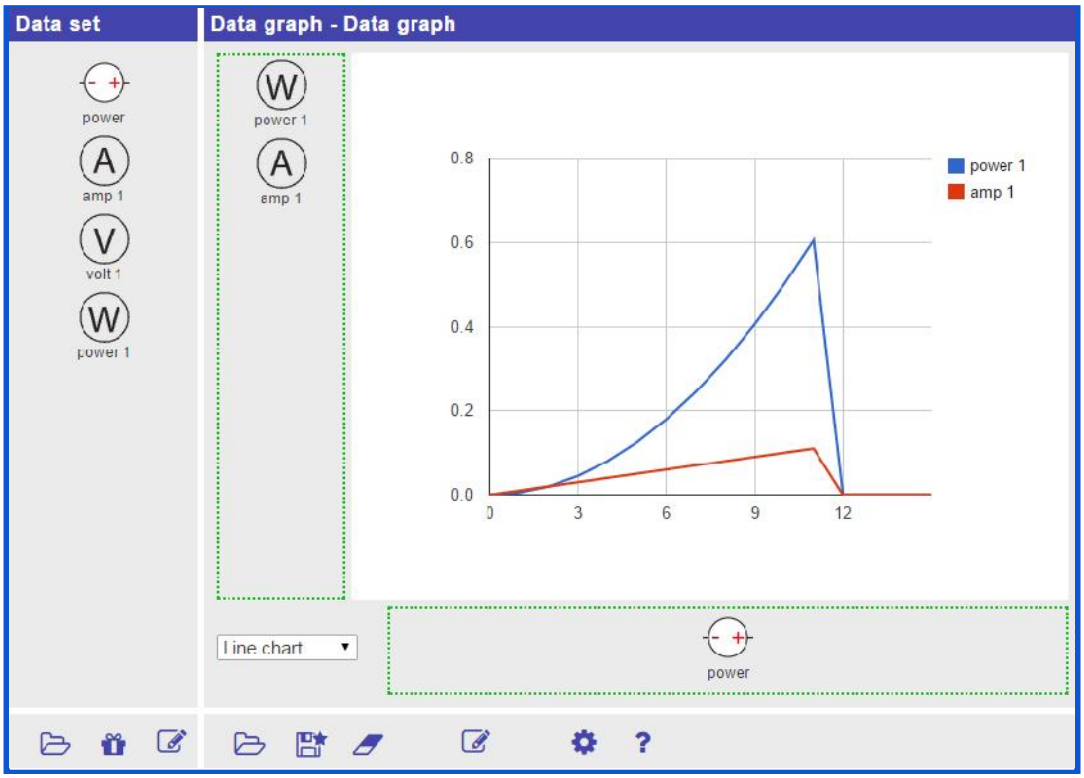


Figure 23. Screenshot of the Data Viewer

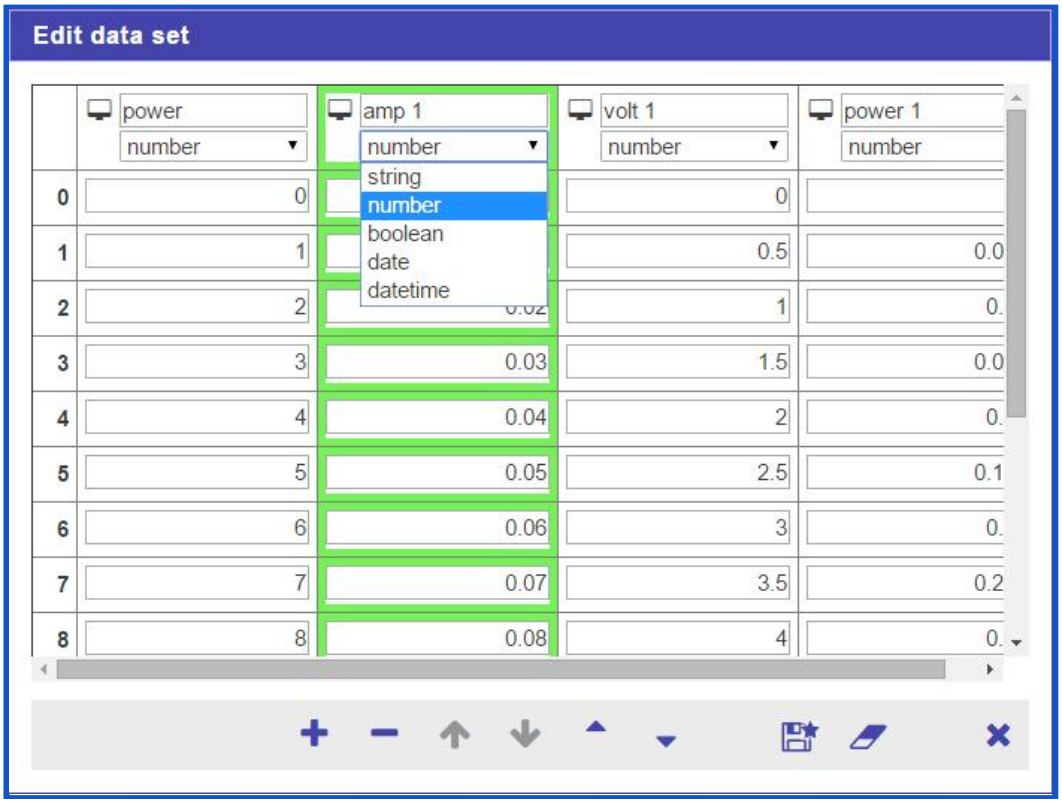


Figure 24. Data set editing feature of the Data Viewer

Changes from previous version:

- The design and color scheme has been modified to match the new Graasp user interface.
- Responsive design compliant with the Graasp style.
- The Data Viewer provides an integrated configuration dialog, which enables a teacher to customize the available features. For this tool, this is limited to the available Help text.
- The Data Viewer is making full use of the Vault storage implementation, which includes an auto-save feature and an auto-load feature.

Quiz Master

URL on the lab repository: http://www.golabz.eu/apps/quizmaster
Supported languages: English, Greek
Predominant phase(s): All
Screenshot: Figure 25

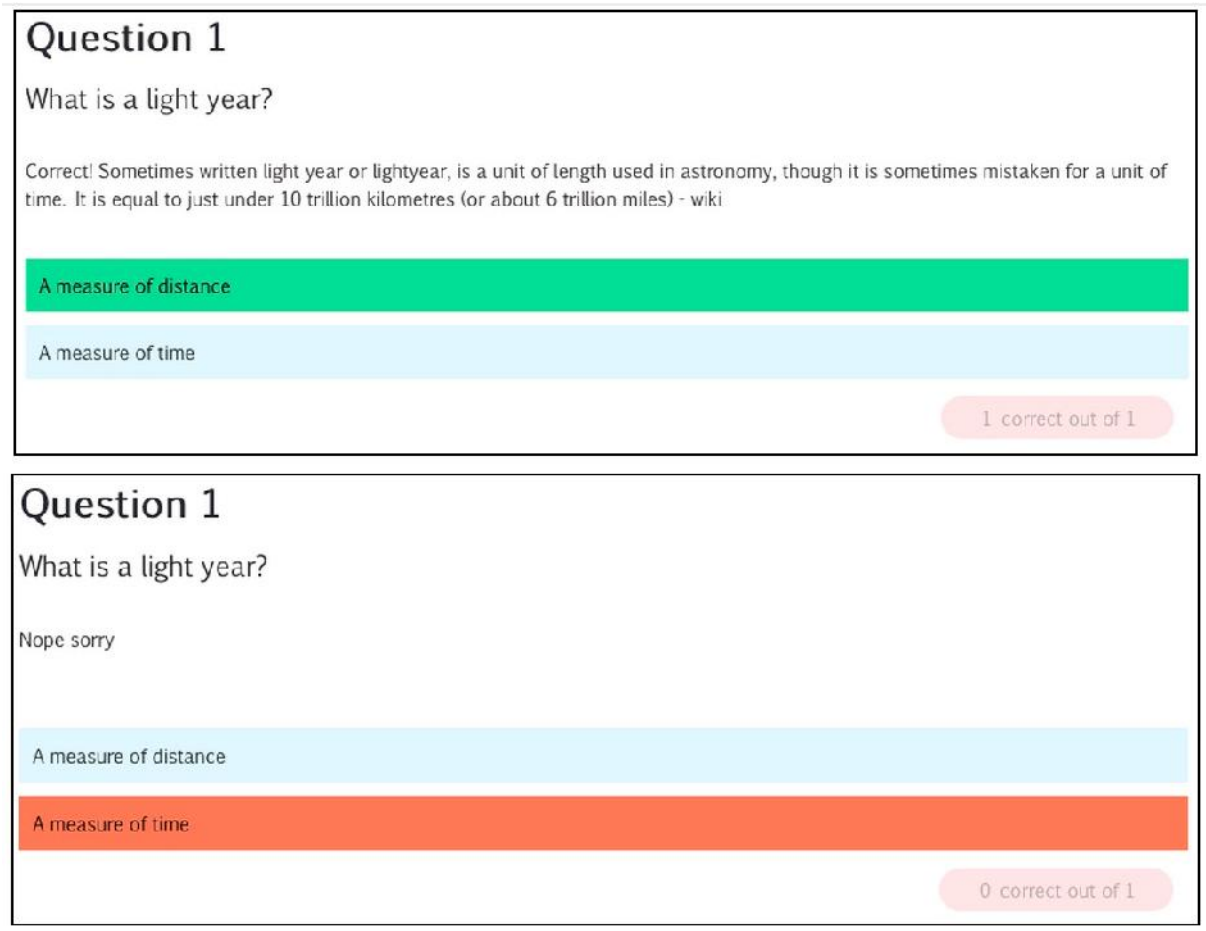


Figure 25. Screenshot of the Quiz Master. Case of student selecting correct (top) and wrong (bottom) answers

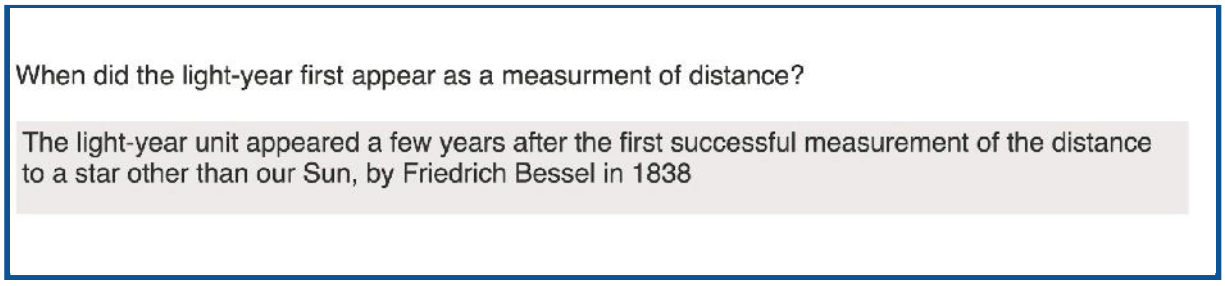
The Quiz Master is a new inquiry learning app that has been designed and developed in the last year following a request from WP1 to support student self-assessment. QuizMaster

presents students with a simple yet compelling multiple choice quiz in the ILS. In more detail it comprises the following features:

- Lets the teachers create a multiple choice quiz with several correct and false answers for each question as well as an optional feedback message that will be displayed to the students after an answer is given.
- Can utilize both Google Spreadsheet and Microsoft Excel files as a quiz source that can easily be created by following a simple format as displayed in the tutorial, which is linked on the app's page at http://www.golabz.eu/apps/quizmaster.
- Utilizes activity streams for tracking users actions.
- Fully utilizes the Vault storage in order to automatically store and load the students' answers.
- It encompases a useful "Teacher's only View" designed to visualize the class answers in detail.
- Responsive design compliant with the Graasp style.

Input Box

URL on the lab repository: http://www.golabz.eu/apps/input-box
Supported languages: English, Greek
Predominant phase(s): All
Screenshot: Figure 26



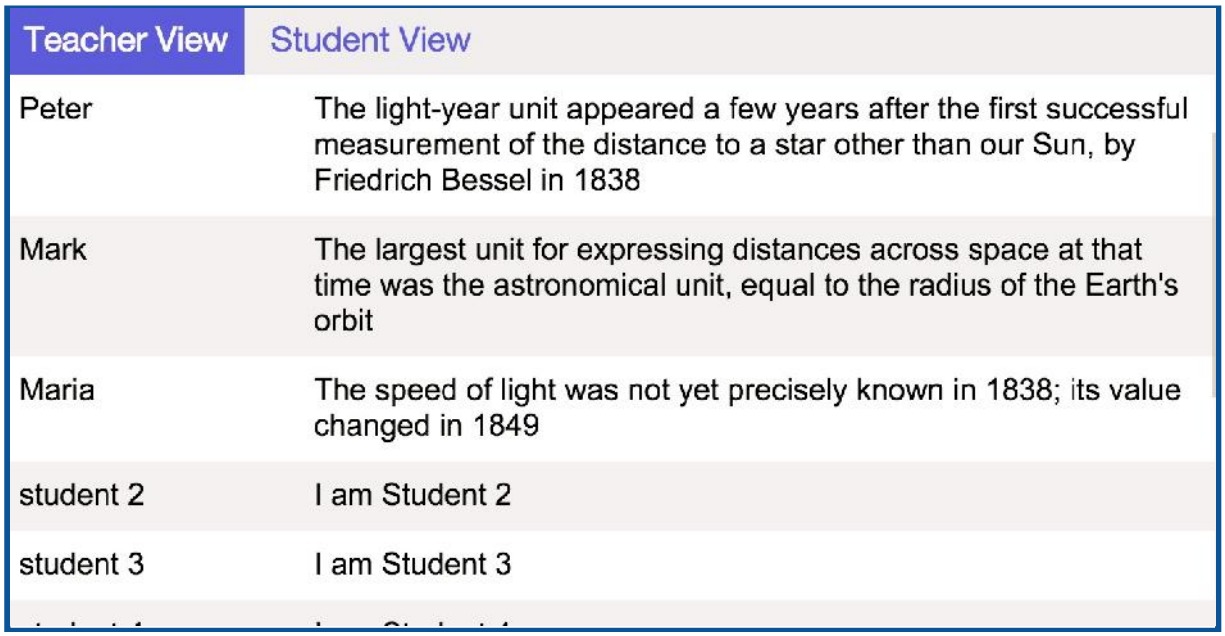


Figure 26. Screenshot of the Input Box. ILS's student view on top and teacher's view in the ILS Platform at the bottom

The Input Box is another newly added app. It supports simple general-purpose note taking and text inputs for the students. In more detail Input Box features:

- Automatically saves and loads students' notes.
- Encompasses a useful "Teacher's View" designed to visualize the class notes in a consolidated view.
- Utilizes activity streams for tracking users actions.
- Fully utilizes the Vault storage in order to automatically store and load data.
- Responsive design compliant with the Graasp style.

Drop File

URL on the lab repository: http://www.golabz.eu/apps/file-drop
Supported languages: English, French, Spanish, and Dutch
Predominant phase(s): Discussion, Conclusion
Screenshot: Figure 27

Short description: The Drop File tool, illustrated in Figures 27.a, 27.b and 27.c, allows students to upload their documents into the ILS by drag-and-drop. As envisioned in D5.3, the uploaded files are stored in the corresponding Vault space of the ILS. The app provides two views depending on the user type: for owners and contributors, the app provides access to all the resources stored in the Vault; for viewers and students, the app only shows resources available in the Vault if the vault is made public.
Usage: Typically, a teacher can add the Drop File tool in the Discussion or Conclusion phases of the ILS. Then, the students will be able to upload their assignment reports into the ILS. The teacher can download student reports through this tool as well. One of the

interesting features of this app is the easy usage in mobile devices. Just by touching the grey area, the app opens a menu to allow the user upload existing or new files (see Figure 27.c).

Technical details: This app uses the document APIs of OpenSocial to create resources in the space where the app has been added. To allow drag-and-drop on the user interface, it uses the Dropzone.js javascript library.

Integration: Currently the Drop File tool stores the uploaded files in the ILS using the credentials of the user logged in the system.

Changes:

- Internationalisation of the app is now possible and several languages are already supported as mentioned above
- Responsive design compliant with the Graasp style
- Visualization criteria: while owners and contributors of the ILS have all their resources stored in the Vault, students can only access them if the Vault is public.

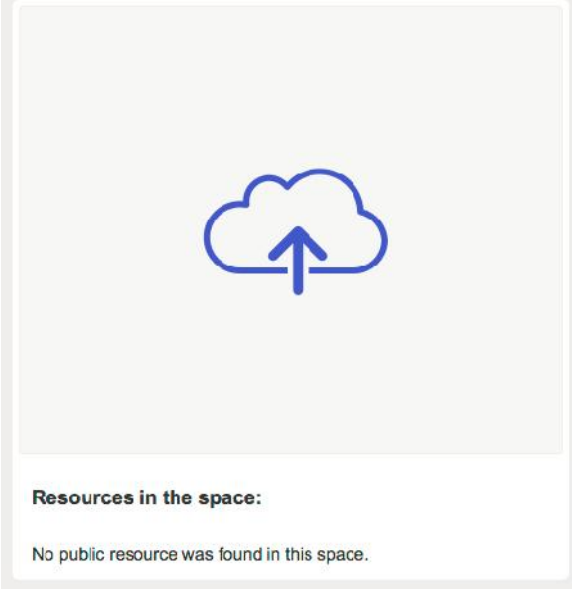


Figure 27.a Example of the File Drop in English where no resource is available for the user

Figure 27.b Example of the File Drop in French where the user has just uploaded a resource

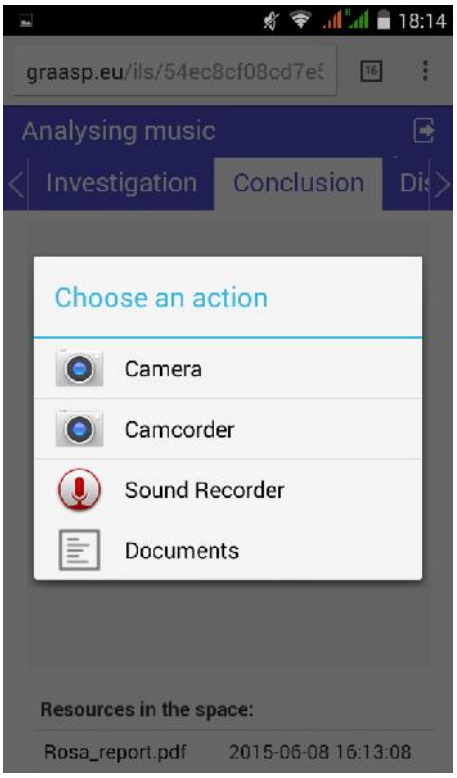


Figure 27.c Screenshot of File Drop in a mobile device

Report Tool (under development)

URL on the lab repository: n/a
Supported languages: n/a
Predominant phase(s): Discussion, Conclusion
Screenshot: Figure 28

Short description: As of writing this deliverable, the Report Tool is under development and in a mock-up / prototype state. The Report Tool is supposed to help learners to reflect and report on their activities in the various inquiry learning phases. To this end, a teacher can personalize this tool through the integrated configuration dialog by specifying (typically reflective) questions, as shown in the mock-up screenshot below. It bears similarities to the Observation Tool in a sense that the learner records his observations/reports in textual form, with the difference that the Report Tool's questions can be configured by the teacher, and it typically serves as an overall reporting facility towards the end of an inquiry learning activity. The Report Tool will integrate and re-use existing resources from previous phases and tools, e.g. from Observation Tools and Hypothesis Scratchpad tools.

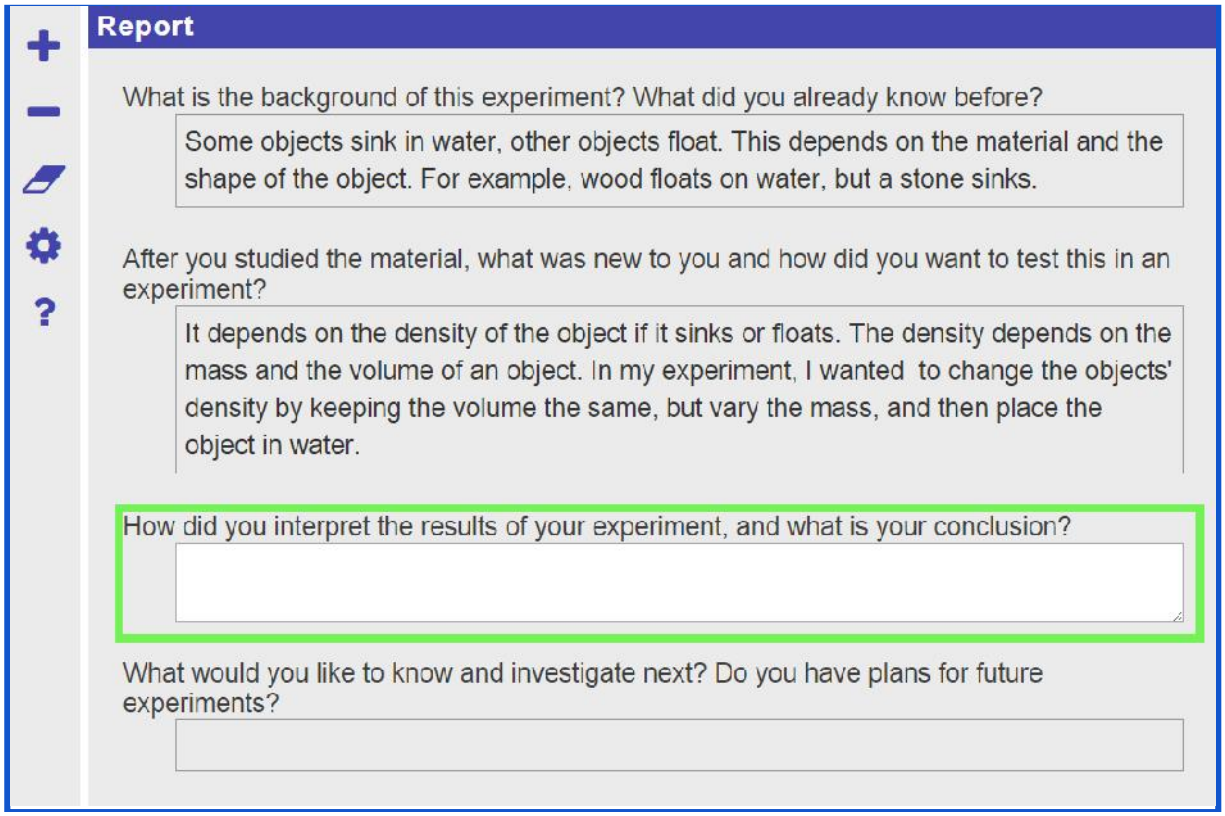


Figure 28. Mock-up of the Report Tool

Resource View App

Resources can now be viewed in the Standalone View simply by dropping them in any phase in Graasp making the resource view app obsolete. Figure 29 shows how a pdf that has been added to the Conceptualisation phase is shown in the Standalone View. Note that some files will be rendered (e.g., pdfs), while others will only be displayed as a link (e.g., xls).

Figure 29. Example of a pdf displayed in the Standalone View.

5 Conclusion

In this software release deliverable, we have presented the implementation work completed for the final release of the personalisation features and the inquiry learning apps. We presented the developments that took place since the initial release presented in D5.3. Since D5.3 we have revised and updated several personalisation features and inquiry learning apps based on input from WP1 and on the continuous feedback received by WP3 documented for instance in D3.2. Also, feedback from studies and experiments conducted in WP8 (G8.3 and D8.3) helped to improve and evaluate the personalisation features and inquiry learning apps.

Whereas D5.3 mainly discussed Internationalisation, in this deliverable, we elaborated on the personalisation features in the authoring process. We also presented the development of the personalisation features for inquiry learning apps through the app configuration feature. This feature, which smoothly integrates features of the App Composer directly in the apps themselves is a major new usability improvement. We will discuss internationalisation of the Portal in its final release in D5.6 on M36.

Changes and extensions to the ILS library, inter-widget communication, and storage facilities have been described. Particularly new, integrated app configuration features have been described, along with a distinct storage mechanism that allows to copy app configuration and personalisation data, while keeping student artefacts private within an ILS.

Finally, the deliverable presented the current state of development of several inquiry learning apps, most of them presented in D5.3, namely the Concept Mapper, the Hypothesis Tool, the Experiment Design Tool, the Conclusion Tool, the Data Viewer app, the Drop File app, and the Resource View app (even though this last one is now integrated in the ILS Platform directly). Furthermore, we have presented two new apps, namely the InputBox and the QuizMaster, which were designed following requests from WP1. WP3 is currently evaluating several of these apps and is expected to provide insights and feedback on their design and usefulness in D3.3 in M36.

As of writing this deliverable, the Reporting Tool has been in a mock-up state and is expected to be fully operational in M36. The Reporting Tool provides means to let students write a structured report of their past activities within the context of an ILS, guided by questions and prompts which can be configured and personalized by the teacher.