
OnTask Documentation

Release 1.1.0

Abelardo Pardo

Apr 22, 2019

Contents

1	Version B.4.0.1	1
1.1	What is OnTask	1
1.2	Installation process	3
1.3	Using OnTask	18
1.4	Use Scenarios	69
1.5	Tutorial	81
1.6	Advanced Features	130

1.1 What is OnTask

Welcome to OnTask, the platform offering instructors and educational designers the capacity to use data to personalize the learner experience.

Learning is complex, highly situated, and requires interacting with peers, instructors, resources, platforms, etc. This complexity can be alleviated providing learners with the right support actions. But this process becomes increasingly complex when the number of learners grows. The larger the number of learners, the more difficult is for instructors to provide support. Typical solutions usually include suggesting or providing generic resources that are only relevant to a subset of the audience.

Learner testimonials

“its not the email itself, but what it represents: the accountability for your learning”

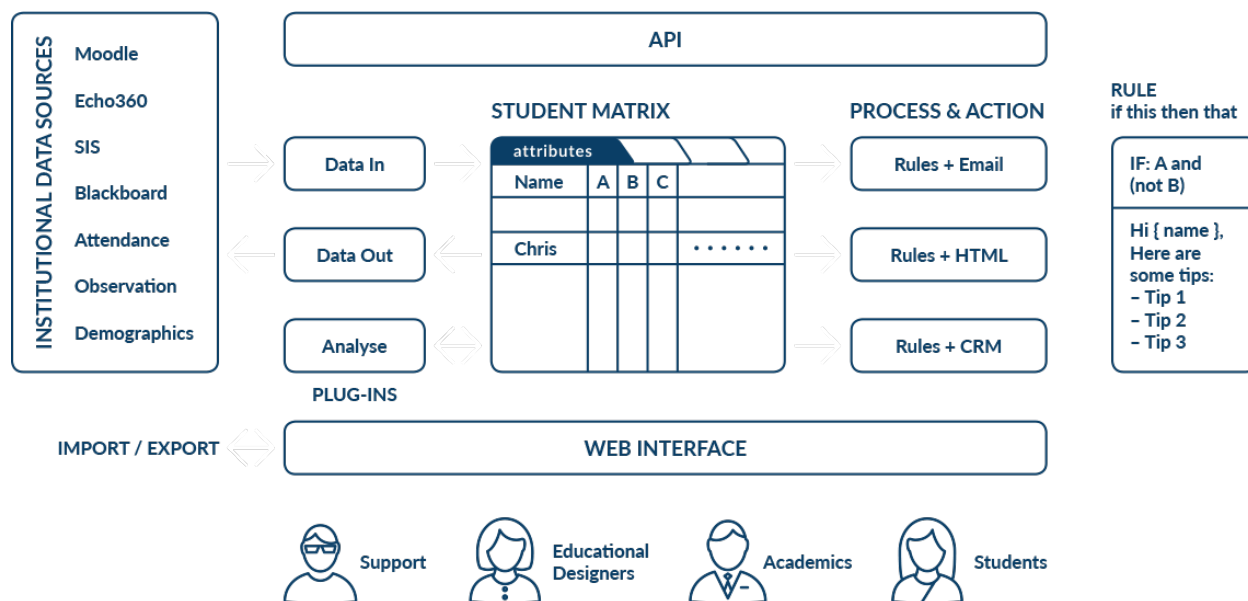
“I think it makes sense as a reflection of the whole semester, oh this is where I struggled with.”

In parallel with this increase in complexity, learning platforms now generate a wealth of data when activities are mediated by technology. This data can be collected and used to help instructors and designers provide a truly personalized experience. Why is this not happening in current platforms? Because the connection between this data and learner support actions is challenging to implement. This is the focus of OnTask: provide instructors and designers with a platform to connect data emerging from learning environments with highly personalized student support actions.

OnTask is a web application that allows instructors and designers to combine data previously captured and stored in a table with a set of rules to personalize the content of a web document. That document can then be sent as an email or made available to each learner. The following picture shows the high level structure of the platform.

Instructor testimonials

“A number of the participants were highly complimentary of the OnTask software tool and several of the teaching staff [...] are actively planning and preparing to use OnTask in the second semester of this academic year.”



“We were also very pleased to support the inclusion of OnTask into the Apereo Incubation process. We have been impressed by the thinking that informs OnTask and would like to see it thrive as an open source solution as we think it has great potential.”

The rest of the document is divided into blocks. The *first block* covers the technical details to download, install and configure the tool. It requires technological expertise and access to the adequate computing facilities (a virtual machine, a server, or similar). The main audience of this part is system administrators and advanced users that want to use the tool within their institution or for their own use.

The *second part* explains how to use OnTask in the context of a learning experience, how to load data, manipulate the table and create the personalized actions. The audience for this part is teachers and designers that need to personalize the interactions they have with learners in a platform offering some data sources.

The *third part* contains a step by step tutorial on how to use OnTask from the instructor’s point of view.

The *fourth part* covers some more advanced functionalities such as an application programming interface (API) that allows other platforms to upload data to OnTask.

1.1.1 Research

There are several platforms that implement similar functionality or follow a similar approach. OnTask has been implemented with numerous ideas initially present in the Student Relationship Engagement System (SRES) and subsequent versions. The common idea among them is the positive impact that personalized communication may have when supporting learners. There are a few scientific publications that document the ideas and processes that inspired the creation of OnTask:

- Pardo, A., Bartimote-Aufflick, K., Buckingham Shum, S., Dawson, S., Gao, J., Gašević, D., ... Vigentini, L. (2018). OnTask: Delivering Data-Informed Personalized Learning Support Actions. *Journal of Learning Analytics*, 5(3), 235-249.
- Pardo, A., Jovanović, J., Dawson, S., Gašević, D., & Mirriahi, N. (2018). Using Learning Analytics to Scale the Provision of Personalised Feedback. *British Journal of Educational Technology*. doi:10.1111/bjet.12592
- Liu, D. Y.-T., Taylor, C. E., Bridgeman, A. J., Bartimote-Aufflick, K., & Pardo, A. (2016). Empowering instructors through customizable collection and analyses of actionable information Workshop on Learning Analytics

for Curriculum and Program Quality Improvement (pp. 3). Edinburgh, UK.

- Liu, D. Y. T., Bartimote-Aufflick, K., Pardo, A., & Bridgeman, A. J. (2017). Data-driven Personalization of Student Learning Support in Higher Education. In A. Peña-Ayala (Ed.), *Learning analytics: Fundamentals, applications, and trends: A view of the current state of the art*: Springer. doi:10.1007/978-3-319-52977-6_5

1.1.2 License

The OnTask software is open source and available under the MIT License.

1.2 Installation process

OnTask is a Web application that manages data about learners to offer them personalized support. For this reason the installation process complies with a set of tight security restrictions. Some of these restrictions lie within the scope of the tool, but others are part of the environment in which the application is installed. We strongly recommend to install OnTask in a web server that uses TTL encryption (HTTPS) to serve all the pages. The application requires exchanging sensitive information about your session with the browser, so the information should be encrypted.

1.2.1 Requirements

OnTask has been developed as a [Django](#) application. Django is a high-level, python-based web framework that supports a rich set of functionality typically required in applications like OnTask. But as with many other applications, OnTask requires a set of additional applications for its execution:

- Python 2.7 **and** Python 3.6
- Django 2.1.4
- Additional Django modules (included in the requirements/base.txt) file
- Redis
- PostgreSQL (version 9.5 or later)

Some of these requirements are handled through Python's package index application [pip](#).

1.2.2 Are you upgrading from a version < 2.8 to 2.8 or later?

If you are upgrading OnTask from a version lower than 2.8 to 2.8 or later, you need to disable the `crontab` used to execute tasks asynchronously from the web server. Starting in version 2.8 those tasks are executed by an application called `celery` that is managed using `supervisor` (see *[Configure the Distributed Task Queue Celery](#)*).

1.2.3 Are you upgrading from version < 4.0 to 4.0 or later?

The upgrade to 4.0 or later requires version 2.7 and 3.6 both installed and available in the system. Django versions 2.0 and later require Python 3 but certain additional libraries used by OnTask have not been fully ported yet and still require the use of Python 2.7. Make sure both versions are available before proceeding to the upgrade.

1.2.4 Installing the required tools

The following installation steps assume that you are deploying OnTask in a production web server capable of serving pages using the HTTPS protocol.

Install and Configure Redis

Django requires Redis to execute as a daemon in the same machine to cache information about the sessions. No specific changes are required in the code, simply have the server running in the background.

1. Download and install [redis](#).

Follow the instructions to configure it to be used by Django.

2. Test that it is executing properly in the background (use the `ping` command in the command line interface).

Install and Configure PostgreSQL

1. Download and install [postgresql](#).
2. Create the role `ontask` with the command `createuser`. The role should be able to create new databases but not new roles and you should define a password for the user (use `createuser --interactive -W`).
3. Adjust the access configuration in `postgresql` (in the configuration file `pg_hba.conf`) to allow the newly created user to access databases locally.
4. Create a new database with name `ontask` with the `createdb` command.
5. Use the client application `psql` to verify that the user has access the newly created database and can create and delete a new table and run regular queries. Test the connection with the following command:

```
psql -h 127.0.0.1 -U ontask -W ontask
```

If the client does not connect to the database, review your configuration options.

Install Python (2.7 and 3.6)

In the following sections we assume that you can open a command line interpreter and you can execute the two python interpreters for versions 2.7 and 3.6.

1. Install [python](#)
2. Verify that the two python interpreters can run and have each the right version (2.7 and 3.6) using the command line interpreter (typically `python --version` or `python3 --version`).
3. Install [pip](#) (the package may be called `python-pip` for Python 2.7 and `python3-pip` for Python 3.6). This tool will be used by both Python and Django to install additional libraries required to execute OnTask.

Download, install and configure OnTask

1. Download or clone a copy of [OnTask](#).
2. Using a command interpreter, go to the OnTask folder and locate a folder inside it with name `requirements`. Verify that it contains the files `base.txt`, `production.txt` and `development.txt`. The first file contains a list of python modules that are required by OnTask. The second is a set of additional modules to run a *production* instance, and the third is the same list if you intend to run a *development* instance.
3. If you plan to run a production instance of OnTask execute the command (you may need administrative privileges to execute this command):

```
pip3 install -r requirements/production.txt
```

Alternatively, if you plan to run a development instance of OnTask, execute the command:


```
pip3 install -r requirements/development.txt
```

This command downloads a set of libraries and modules and installs them as part of the python libraries in the system.

At this point you have the major modules in place. The next steps include the configuration of the Django environment to run OnTask. If you plan to install a **development** instance of OnTask, using a plain text editor (nano, vim, Emacs or similar) in a command line interpreter, open the file `manage.py` in the `src` folder of the project. Modify line 10 replacing the value `"ontask.settings.production"` by `"ontask.settings.development"`. Save and close the file.

Using the same plain text editor create a file with name `local.env` in the folder `src/ontask/settings` with the following content (note there is no space between variable names and the equal sign):

```
DEBUG=False
TIME_ZONE=[YOUR LOCAL PYTHON TIME ZONE]
BASE_URL=''
DOMAIN_NAME=[YOUR DOMAIN NAME]
SHOW_HOME_FOOTER_IMAGE=True
# syntax: DATABASE_URL=postgres://username:password@127.0.0.1:5432/database
DATABASE_URL=postgres://[PSQLUSERNAME]:[PSQLPWD]@127.0.0.1:5432/ontask
REDIS_URL=[YOUR REDIS URL]
SECRET_KEY=
#
# LTI
#
LTI_OAUTH_CREDENTIALS=test=secret,test2=reallysecret
```

1. Open a command interpreter and execute the following python command:

```
python -c 'import tzlocal; print(tzlocal.get_localzone().zone)'
```

Replace `[YOUR LOCAL PYTHON TIME ZONE]` in the `local.env` file by the description of your time zone produced by the previous command.

2. If OnTask is going to be served from a location different from the root of your server (for example `myhost.com/ontask`), then modify the value of the variable `BASE_URL` with the suffix that should follow the domain name (in the example, `/ontask`).
3. Modify the line starting with `DOMAIN_NAME=` and change the field `[YOUR DOMAIN NAME]` with the domain name of the machine hosting OnTask.
4. If you want to disable the appearance of the banner image in the login page set the value of the variable `SHOW_HOME_FOOTER_IMAGE` to `False`.
5. Modify the line starting with `DATABASE_URL=` and change the field `[PSQLUSERNAME]` with the name of the Postgresql user created in the previous step (the one that could access the ontask database and run queries). If you decided to use a different name for the database, adjust the last part of the line accordingly (replace *ontask* by the name of your database).
6. Replace the string `[YOUR REDIS URL]` with the URL where Redis can be accessed. This is typically something similar to `redis://127.0.0.1:6379/1`.
7. Open a command interpreter and execute the following python command:

```
python3 -c 'import random; import string; print("".join([random.SystemRandom().
↳choice(string.digits + string.ascii_letters + string.punctuation) for i in_
↳range(100)]))'
```

Copy the long string produced as output and add it at the end of the last line of the file `local.env`. It should look something like (with different content after the equal sign):

```
SECRET_KEY=4o93jf0572094jv...
```

8. Modify the line starting with `LTI_OAUTH_CREDENTIALS` and include a comma-separated list of pairs `key=secret` for LTI authentication. See the section [Authentication](#) for more details about this type of authentication.
9. Create a new folder with name `logs` in the OnTask top folder (next to the `requirements` folder). This folder **is different** from the folder with the same name in the `src` folder.
10. If at some point during the following steps you want to reset the content of the database, run the commands `dropdb` and `createdb` explained in [Install and Configure PostgreSQL](#).
11. Execute the following command from the `src` folder to create the database internal structure:

```
python3 manage.py migrate
```

A few messages should appear on the screen related to the initialization of the database.

12. Execute the following command to upload to the platform some initial data structures:

```
python3 manage.py runscript -v1 --traceback initial_data
```

The command should run without any error or exception. If you need to create additional users before deploying the platform, read the section [Creating users in Bulk](#).

13. Execute the command to create a superuser account in OnTask:

```
python3 manage.py createsuperuser
```

Remember the data that you enter in this step so that you use it when you enter OnTask with your browser.

14. Go to the `docs_src` folder to generate the documentation. Make sure this folder contains the sub-folders with name `_static` and `_templates`. Execute the command:

```
make clean html copy_to_docs
```

The documentation is produced by the `sphinx-doc` application and generates the directory `_build` which is then created to the folder `../docs` folder.

15. From the `src` folder execute the following command to collect and install the static content:

```
python3 manage.py collectstatic
```

16. If you are running a production instance, execute the following command to check the status of the platform:

```
python3 manage.py check --deploy
```

The command should print just one warning about the configuration variable `X_FRAME_OPTIONS`. If you are running a development instance, you will get various additional warning that are derived most of them from running the instance without HTTPS.

17. Execute the following command to start the OnTask server:

```
python3 manage.py runserver
```

If there are no errors, the message on the screen should say that your server is running in the url `127.0.0.1:8000`. However, if you open your browser in that URL, an error will be shown. This error is normal and it is because

the production version requires the pages to be served through SSL with a valid certificate in a conventional server.

18. If OnTask is going to be accessed through a web server like Apache or Nginx, stop the application and configure the web server accordingly.
19. If you want to use the server in development mode through the URL `127.0.0.1:8000` you have to perform two more steps. First, edit the file `manage.py` and change these three lines to look like:

```
os.environ.setdefault("DJANGO_SETTINGS_MODULE",
                      "ontask.settings.development")
```

Second, execute the following command from the `src` folder:

```
pip3 install -r requirements/development.txt
```

Now, the command:

```
python3 manage.py runserver
```

will start the server in the URL `127.0.0.1:8000` and you should be able to access it normally with the browser.

Warning

The development version of OnTask is **not suited** to be used in production because it disables several security features. Make sure you only deploy a **production** version.

Configure the Distributed Task Queue Celery

There are various tasks that need to be executed by OnTask outside the web server. The solution adopted is to use [Celery](#), [Supervisor](#) (a process control system) and [Redis](#). Redis has been configured in a previous step. This section explains how to set up the distributed task queue and make sure it is continuously executing in parallel with the web server.

1. Install the application `supervisor` using **pip2**. This application makes sure the application Celery is continuously running in the background and in communication with the server. As of now, the application can only run through Python 2 (an upgrade in the future may remove this anomaly). For this reason, the package has to be installed with the following command (using Python 2.7):

```
pip2 install supervisor
```

2. Check that the binaries `supervisord`, `supervisorctl` and `celery` are installed in your system.
3. Go to the folder `supervisor` in the top of the project and edit the file `supervisor.conf`.
4. The file configures `supervisord` to run in the background and prepare two sets of processes for OnTask. You have two options to use this file:
 - a) Use environment variables.

The file uses internally the value of two environment variables:

- `PROJECT_PATH`: Full path to the root of the project (the top folder containing the file `LICENSE`).
- `CELERY_BIN`: Full path to the executable `celery` in your system (typically `/usr/local/bin/celery` or similar).

- Set these variables in your environment to the correct values and make sure they are properly exported and visible when running other commands. For example, in `bash`, this operation would be achieved by two commands similar to:

```
$ export PROJECT_PATH=/full/path/to/OnTask/root/folder
$ export CELERY_BIN=/full/path/to/celery/executable
```

b) Change the file `supervisor.conf`.

- replace any appearance of the string `%(ENV_PROJECT_PATH)s` by the full path to the project folder.
- replace any appearance of the string `%(ENV_CELERY_BIN)s` by the full path to the `celery` binary program.

4. Start the process control system with the command:

```
$ supervisord -c supervisor.conf
```

The command starts the process control application `supervisord` which executes a set of processes in the background.

5. Check that the process control system is working with the command (executed from the `supervisor` folder):

```
$ supervisorctl -c supervisor.conf status
```

The output of this command should show a message similar to:

ontask-beat-celery	RUNNING	pid 28579, uptime 1 day, 0:07:36
ontask-celery	RUNNING	pid 28578, uptime 1 day, 0:07:36

If the status of the two processes is `STARTING` wait a few seconds and execute the command again. The names `ontask-beat-celery` and `ontask-celery` are the names of the two processes that OnTask uses for asynchronous task execution.

You may use this command to check if `supervisord` is still running. The application is configured to write its messages to the file `celery.log` in the `logs` folder at the top of the project.

6. If you are upgrading OnTask from a previous version (less than 2.8), you need to edit the `crontab` entry and remove the command to execute the script `scheduler_script.py`.

1.2.5 Upgrading OnTask

If you have OnTask already configured and running, here are the steps to follow to upgrade to a new version. If you are upgrading from a version below 2.8 to 2.8 or higher read [Configure the Distributed Task Queue Celery](#).

If you are upgrading from a version below 4.0, make sure that:

- Versions 2.7 and 3 of Python are properly installed.
- The library `Celery` is installed as part of the Python 3 system and the binary is executed using Python 3.
- The library `supervisord` is installed as part of the Python 2.7 system and the binary is executed using Python 2.

After verifying the previous requirements, proceed with the following steps.

- Create a backup of the database to be able to restore the state of the tool before the upgrade process.
- Stop the apache web server.

- Open a terminal and use a command interpreter to execute the following commands.
- Place the interpreter in the project folder (the one with the folder `src` in it)
- Pull the code for the new version from the repository:

```
git pull
```

- Refresh the list of requirements:

```
pip install -r requirements/production.txt
```

- Go to the sub-folder containing the tool documentation:

```
cd docs_src
```

- Re-create the tool documentation and place it in the appropriate folder:

```
make clean html copy_to_docs
```

- Go to the sub-folder containing the apps:

```
cd ../src
```

- Collect all files to be served statically:

```
python manage.py collectstatic
```

- Apply the migrations to the database:

```
python manage.py migrate
```

- Check that the configuration is ready to run:

```
python manage.py check --deploy
```

- Restart the supervisord configuration:

```
supervisorctl -c ../supervisor.conf reload
```

- Restart the apache web server and check the new version is properly installed.

1.2.6 The Administration Pages

OnTask uses the administration pages offered by Django. The account created with the command `createsuperuser` has complete access to those pages through a link in the upper right corner of the screen.

These pages offer access to several important operations:

- The elements of each of the models stored in the database (workflows, actions, conditions, columns, etc). Each model has its corresponding page allowing the creation, update and deletion of any object.
- The user information. This is a special model representing the users, their name, credentials, etc. The platform allows the creation of user accounts.
- The group information. The platform differentiates users based on groups. Each group has different functionalities.

Once the instance is running, visit these pages and configure the platform to your needs.

1.2.7 Production Deployment

Once OnTask is executing normally, you may configure a web server (nginx, apache or similar) to make it available to a community of users. The instructions to make such deployment are beyond the scope of this manual but are available through the corresponding manual pages of these applications.

1.2.8 Authentication

OnTask comes with the following authentication mechanisms: IMS-LTI, `REMOTE_USER` variable, basic authentication, and LDAP. The first three (IMS-LTI, `REMOTE_USER` and basic authentication) are enabled by default and used in that order whenever an unauthenticated request is received. It follows a brief description of how to configure them.

- **IMS Learning Tools Interoperability (IMS-LTI).** LTI is a standard developed by the IMS Global Learning Consortium to integrate multiple tools within a learning environment. In LTI terms, OnTask is configured to behave as a *tool provider* and assumes a *tool consumer* such as a Learning Management System to invoke its functionality. Any URL in OnTask can be given to the LTI consumer as the point of access.

OnTask only provides two points of access for LTI requests coming from the consumer. One is the URL with suffix `/lti_entry` and the second is the URL provided by the actions to serve the personalized content (accessible through the **Actions** menu).

To allow LTI access you need:

- 1) A tool consumer that can be configured to connect with OnTask. This type of configuration is beyond the scope of this manual.
- 2) A set of pairs key,value in OnTask to be given to the tool consumers so that together with the URL, they are ready to send the requests. The key/value pairs need to be included as an additional variables in the file `local.env` in the folder `src/ontask/settings` together with other local configuration variables. For example,

```
LTI_OAUTH_CREDENTIALS=key1=secret1,key2=secret2
```

If you change the values of this variable, you need to restart the server so that the new credentials are in effect.

This authentication has only basic functionality and it is assumed to be used only for learners (not for instructors).

- **REMOTE_USER.** The second method uses the variable `REMOTE_USER` that is assumed to be defined by an external application. This method is ideal for environments in which users are already authenticated and are redirected to the OnTask pages (for example, using SAML). If OnTask receives a request from a non-existent user through this channel, it automatically and transparently creates a new user in the platform with the user name stored in the `REMOTE_USER` variable. OnTask relies on emails to identify different user names, so if you plan to use this authentication method make sure the value of `REMOTE_USER` is the email.

Additionally, this mode of authentication will be enforced in all requests reaching OnTask. However, this configuration prevents the recording of email reads. Read the section [Email Configuration](#) to configure the server to allow such functionality to be properly configured.

- **Basic authentication.** If the variable `REMOTE_USER` is not set in the internal environment of Django where the web requests are served, OnTask resorts to conventional authentication requiring email and password. These credentials are stored in the internal database managed by OnTask.

The API can be accessed using through token authentication. The token can be generated manually through the user profile page. This type of authentication may need some special configuration in the web server (Apache or similar) so that the `HTTP_AUTHORIZATION` header is not removed.

LDAP Authentication

OnTask may also be configured to use LDAP to authenticate users. This is done through the external package `django-auth-ldap`. In its current version, this authentication mode cannot be combined with the previous ones (this requires some non-trivial code changes). The following instructions describe the basic configuration to enable LDAP authentication. For more details check the [documentation of the django-auth-ldap module](#).

- Stop OnTask (if it is running)
- Make sure your server has installed the development files for OpenLDAP. In Debian/Ubuntu, the required packages are:

```
libsasl2-dev python-dev libldap2-dev libssl-dev
```

In RedHat/CentOS:

```
python-devel openldap-devel
```

- Install the module `django-auth-ldap`
- Edit the configuration file `local.env` and add the following two variable definitions:

```
AUTH_LDAP_SERVER_URI=[uri pointing to your ldap server]
AUTH_LDAP_PASSWORD=[Password to connect to the server]
```

- Edit the file `src/ontask/settings/base.py` and uncomment the lines that import the `ldap` library (`import ldap`) and the lines that import three methods from the `django_auth_ldap.config` module (`LDAPSearch`, `GroupOfNamesType` and `LDAPGroupQuery`)
- Locate the section in the file `src/ontask/settings/base.py` that contains the variables to configure *LDAP AUTHENTICATION*.
- Uncomment the ones needed for your configuration. Make sure all the information is included to connect to the server, perform the binding, search, and if needed, assign fields to user and group attributes.
- Locate the variable `AUTHENTICATION_BACKENDS` in the same file.
- Comment the lines referring to the back-ends `LTIAuthBackend` and `RemoteUserBackend`.
- Uncomment the line referring to `LDAPBackend`.
- Make sure the LDAP server contains the data about the users in the right format
- Start the OnTask server.

1.2.9 Email Configuration

OnTask relies on the functionality included in Django to send emails from the application. The following variables must be defined in the file `local.env`:

- `EMAIL_HOST`: Host providing the SMTP service,
- `EMAIL_PORT`: Port to communicate with the host,
- `EMAIL_HOST_USER`: User account to log into the email host,
- `EMAIL_HOST_PASSWORD`: Password for the account to log into the email host,
- `EMAIL_USE_TLS`: Boolean stating if the communication should use TLS,
- `EMAIL_USE_SSL`: Boolean stating if the communication should use SSL,

- `EMAIL_ACTION_NOTIFICATION_SENDER`: Address to use when sending notifications,
- `EMAIL_BURST`: Number of consecutive emails to send before pausing (to adapt potential throttling of the SMTP server)
- `EMAIL_BURST_PAUSE`: Number of seconds to wait between bursts.

An example of configuration is:

```
EMAIL_HOST=smtp.yourinstitution.org
EMAIL_PORT=334
EMAIL_HOST_USER=mailmaster
EMAIL_HOST_PASSWORD=somepassword
EMAIL_USE_TLS=False
EMAIL_USE_SSL=False
EMAIL_ACTION_NOTIFICATION_SENDER=ontaskmaster@yourinstitution.org
EMAIL_BURST=500
EMAIL_BURST_PAUSE=43200
```

Set these variables in the configuration file to the appropriate values before starting the application. Make sure the server is running in **production mode**. The development mode is configured to **not send** emails but show their content in the console instead.

Tracking Email Reads

If OnTask is deployed using SAML, all URLs are likely to be configured to go through the authentication layer. This configuration prevents OnTask from receiving the email read confirmations. In this case, the web server needs to be configured so that the SAML authentication is removed for the URL `trck` (the one receiving the email read tracking). In Apache, this can be achieved by the following directive:

```
<Location /trck>
  Require all granted
</Location>
```

If OnTask is not served from the root of your web server, make sure you include the absolute URL to `trck`. For example, if OnTask is available through the URL `my.server.com/somesuffix/ontask`, then the URL to use in the previous configuration is `my.server.com/somesuffix/ontask/trck`.

1.2.10 Canvas Email Configuration

OnTask allows to send personalized emails to the user inbox in an instance of a [Canvas Learning Management System](#) using its API. Configuring this functionality requires permission from Canvas to access its API using OAuth2 authentication. Once this authorization is obtained, the following variables need to be defined in the file `local.env` configuration file:

- `CANVAS_INFO_DICT`: A dictionary with elements pairs containing the identifier for a Canvas instance that will be shown to the user and a dictionary with the following configuration parameters:
 - `domain_port`: A string containing the domain and port (if needed) of the Canvas host.
 - `client_id`: This value is provided by the administrator of the Canvas instance once permission to use the API has been granted.
 - `client_secret`: This value is provided together with the `client_id` once the permission to use the API is granted. It is typically a large random sequence of characters.
 - `authorize_url`: URL template to access the first step of the authorization. This is usually `https://{0}/login/oauth2/auth`. The string `{0}` is replaced internally with the value of `domain_port`.

- `access_token_url`: URL template to access the token. This is usually `https://{0}/login/oauth2/token`. The string `{0}` is replaced internally with the value of `domain_port`.
- `conversation_URL`: Similar to the previous two values, it is the entry point in the API to create a conversation (equivalent to send an email). This is usually `https://{0}/api/v1/conversations`. The string `{0}` is replaced internally with the value of `domain_port`.
- `aux_params`: A dictionary with additional parameters. The dictionary may include a value for the key `burst` to limit the number of consecutive API invocations (to prevent throttling) and a value for the key `pause` with the number of seconds to separate bursts. Here is an example of the definition of this variable in the `local.env` file:

```
CANVAS_INFO_DICT={"Server one": {"domain_port": "yourcanvasdomain.edu",
↪ "client_id": "10000000000001", "client_secret":
↪ "YZnGjbkopt9MpSq2fujUOgbeVZ8NdkdCeGF2ufhWZdBKAZvNCuuTOWXHotSWMu6X",
↪ "authorize_url": "http://{0}/login/oauth2/auth", "access_token_url": "http:/
↪/{0}/login/oauth2/token", "conversation_url": "http://{0}/api/v1/
↪conversations", "aux_params": {"burst": 10, "pause": 5}}}
```

- `CANVAS_TOKEN_EXPIRY_SLACK`: The number of seconds to renew a token before it expires. For example, if the variable is 300, any API call performed with a token five minutes before it expires will prompt a token refresh. Here is an example of such definition in `local.env`:

```
CANVAS_TOKEN_EXPIRY_SLACK=300
```

After defining these variables, restart the application for the values to be considered. To test the configuration open a workflow, create an action of type `Personalized canvas email` and email those messages.

1.2.11 Plugins

OnTask allows also the inclusion of arbitrary Python modules to execute and transform the data stored in a workflow. The Python code in the plugins is executed the same interpreter and execution environment as the rest of the platform. Thus, **use this functionality to execute only code that is fully trusted**. There is nothing preventing a plugin to run malicious code (think `system.exec('rm -rf /')`, so use at your own risk. To configure the execution of plugins follow these steps:

1. Create a folder at any location in your instance of OnTask to store the Python modules. OnTask assumes that each directory in that folder contains a Python module (that is, a folder with a file `__init__.py` inside).
2. Open the administration page of OnTask as superuser and go to the section with title *Data Upload/Merge Operations*.
3. Select the *Preferences* section.
4. Modify the field *Folder where plugins are installed* to contain the absolute path to the folder created in your systems.
5. Make sure that the Python interpreter that is currently executing the Django code is also capable of accessing and executing the code in the plugin folder.
6. Restart the server to make sure this variable is properly updated.
7. To create a new plugin first create a folder in the plugin space previously configured.
8. Inside this new folder create a Python file with name `__init__.py`. The file has to have a structure a shown in the following template:

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals, print_function

import pandas as pd

# The field class_name contains the name of the class to load to execute the
# plugin.
class_name = 'OnTaskTestPlugin'

class OntaskTestPlugin(object):
    """
    Example of a class that implements the OnTask plugin interface. The
    objects of this class have to provide the following elements:

    1. Field name: Plugin name show to the users.

    2. Field description_txt: A string with the detailed description of what the
    plugin does

    3. Field input_column_names: A potentially empty list of column names
    (strings). If the list is empty, the columns are selected by the user at
    execution time.

    4. Field output_column_names: Non empty list of names (strings) of the
    columns to be used for the output of the transformation.

    5. Field parameters: an optionally empty list with tuples with the following
    structure:

        ('name', type, [list of allowed values], initial value, help_text)

    These elements will be requested from the user before executing the
    plugin through a form. The conditions on these values are:

    - name must be a string
    - type must be a string equal to "integer", "double", "string",
      "datetime" or "boolean".
    - The list of values is to restrict the
      possible values
    - The initial value must be of the type specified by the second
      element.
    - Help_text a string to show as help text

    6. Method "run" that receives:
        - a pandas data frame with the data to process
        - a string with the name of the key column that will be used to merge
          the result.
        - A dictionary of pairs (name, value) with the parameters described in
          the previous element.

    and returns a result Pandas data frame. This frame must have one
    column with the key column name provided so that it can be properly
    merged with the existing data.
    """

    def __init__(self):
```

(continues on next page)

(continued from previous page)

```

self.name = 'Plugin Template'
self.description_txt = 'Example of plugin description'
self.input_column_names = list()
self.output_column_names = ['RESULT 1', 'RESULT 2']
self.parameters = [
    ('param string', 'string', ['v1', 'v2'], 'v1', 'help param string'),
    ('param integer', 'integer', [], None, 'help param integer'),
    ('param double', 'double', [1.2, 2.2, 3.2], None,
     'help param double'),
    ('param boolean', 'boolean', [], True, 'help param boolean'),
    ('param datetime', 'datetime', [], '2018-05-25 18:03:00+09:30',
     'help param datetime'),
    ('param datetime2', 'datetime',
     [],
     '2018-05-25 18:03:00+09:30',
     'help param datetime'),
]

def run(self, data_frame, merge_key, parameters=dict):
    """
    Method to overwrite. Receives a data frame with a number of columns
    stipulated by the num_column_input pair, the name of a key column and a
    dictionary with parameters of the form name, value.

    Runs the algorithm and returns a pandas data frame structure that is
    merged with the existing data frame in the workflow using the merge_key.

    :param data_frame: Input data for the plugin
    :param merge_key: Name of the column key that will be used for merging
    :param parameters: Dictionary with (name, value) pairs.

    :return: a Pandas data_frame to merge with the existing one (must
    contain a column with name merge_key)
    """

    # Extract the key column from the given data frame
    result = pd.DataFrame(data_frame[merge_key])

    # Process the given data and create the result
    result[self.output_column_names[0]] = 1
    result[self.output_column_names[1]] = 2

    return result

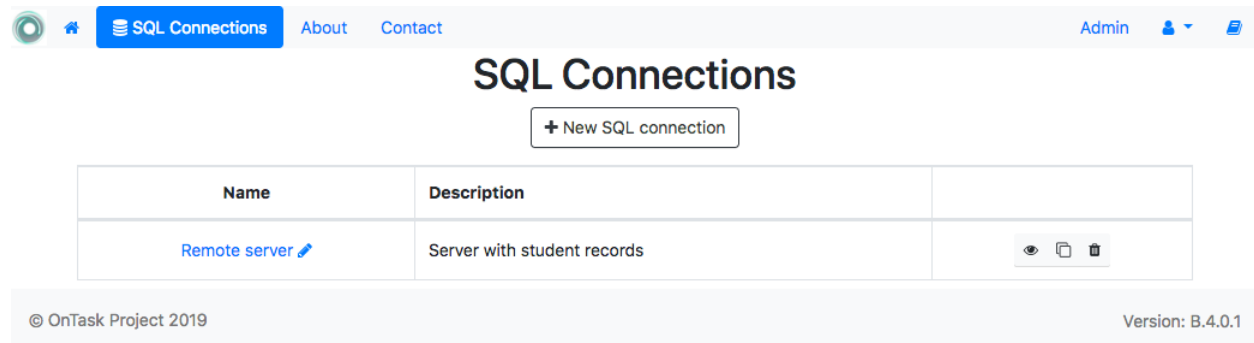
```

9. The menu *Dataops* at the top of the platform includes the page *Transform* that provides access to the plugins and its invocation with the current workflow.

1.2.12 SQL Connections

One of the key functionalities of OnTask is to be able to merge data from multiple sources. Section [Data](#) describes the functionality available to perform these operations. Some of them, however, require special configuration from the tool administrator. This is the case when uploading and merging data from a remote database that allows SQL connections. These connections must be first defined by the administrator and are then available to the instructors.

The screen to manage these connections is accessed clicking in the item *SQL Connections* at the top menu bar. This link is only available for those users with the administration role.



Each connection can be defined with the following parameters:

Name (required) Name of the connection for reference purposes within the platform. This name must be unique across the entire platform.

Description A paragraph or two explaining more detail about this connection.

Type (required) Type of database connection to be used. Typical types include *postgres*, *mysql*, etc.

Driver Driver to be used for the connection. OnTask assumes that these drivers are properly installed and available to the underlying Python interpreter running Django.

User User name to connect to the remote database.

Requires password Flag denoting if the connection requires password. If it does, the password will be required at execution time. This feature allows OnTask to avoid storing DB passwords.

Host Host name or IP storing the remote database

Port Port to use to connect to the remote host

DB Name (required) Name of the remote database

Table (required) Name of the table stored in the remote database and containing the data to upload/merge

Once a connection is defined, as described in *SQL connection*, all the data in the table will be accessed and loaded/merged into the current workflow.

The operations allowed for each connection are:

Edit Change any of the parameters of the connection

Clone Create a duplicate of the connection (useful to reuse configuration parameters)

Delete Remove the connection from the platform.

1.2.13 Creating users in Bulk

OnTask offers the possibility of creating users in bulk through given the data in a CSV file through the following steps:

1. Create a CSV file (plain text) with the initial line containing only the word `email` (name of the column). Include then one email address per user per line. You may check the file `initial_learners.csv` provided in the folder `src/scripts`.
2. From the `src` folder run the command:

Edit SQL connection



Name*

Remote server

Description

Server with student records

Type*

mysql

Postgresql, Mysql, etc.

Driver

Driver implementing the DBAPI

User

remote_db_user

☐ Requires password?

Host

dbserver.bogus.com

Port



DB name*

```
$ python manage.py runscript initial_data --script-args "-d scripts/initial_learners.csv"
```

If you have the user emails in a file with a different column name, you may provide the script that name (instead of the default email using the option `-e`:

```
$ python manage.py runscript initial_data --script-args "-d -e your_email_column_name scripts/initial_learners.csv"
```

If you want to create user accounts for instructors, you need to specify this with the option `-i` in the script:

```
$ python manage.py runscript initial_data --script-args "-d -e your_email_column_name -i scripts/initial_learners.csv"
```

1.3 Using OnTask

Contents

- *Workflow*
- *Data*
 - *Upload Data*
 - * *Upload CSV Files*
 - * *Upload Excel Files*
 - * *Upload Google Spreadsheet Files*
 - * *SQL connection*
 - * *Confirmation step to upload data*
 - *Data Merge*
 - * *Step three: select keys and merge option*
 - * *Step four: verify upcoming changes*
- *Table*
 - *Table Views*
- *Actions*
 - *Personalized Text*
 - * *Using column values, attributes and conditions in a Personalized Text*
 - *Surveys*
 - *Personalized JSON Object*
 - *Personalized Canvas Email*
 - *Running actions*
 - * *Sending personalized emails (Personalized Text Actions)*
 - * *Making personalized content available to learners*

- * *Running a survey*
- * *Sending personalized emails in Canvas*
- * *Sending personalized JSON objects to another platform*
- *Creating a ZIP file with the personalized text*
 - * *Uploading feedback files for a Moodle Assignment*
- *Workflow Settings*
 - *Workflow Operations*
 - *Column Operations*
 - *Scheduled Actions*
 - * *Scheduling a Personalized Text Action*
 - * *Scheduling a Personalized JSON Action*
 - * *Table with Scheduled Actions*
 - *View logs*
- *Plugins: Transforming the data with your own code*
 - *Plugin requirements*

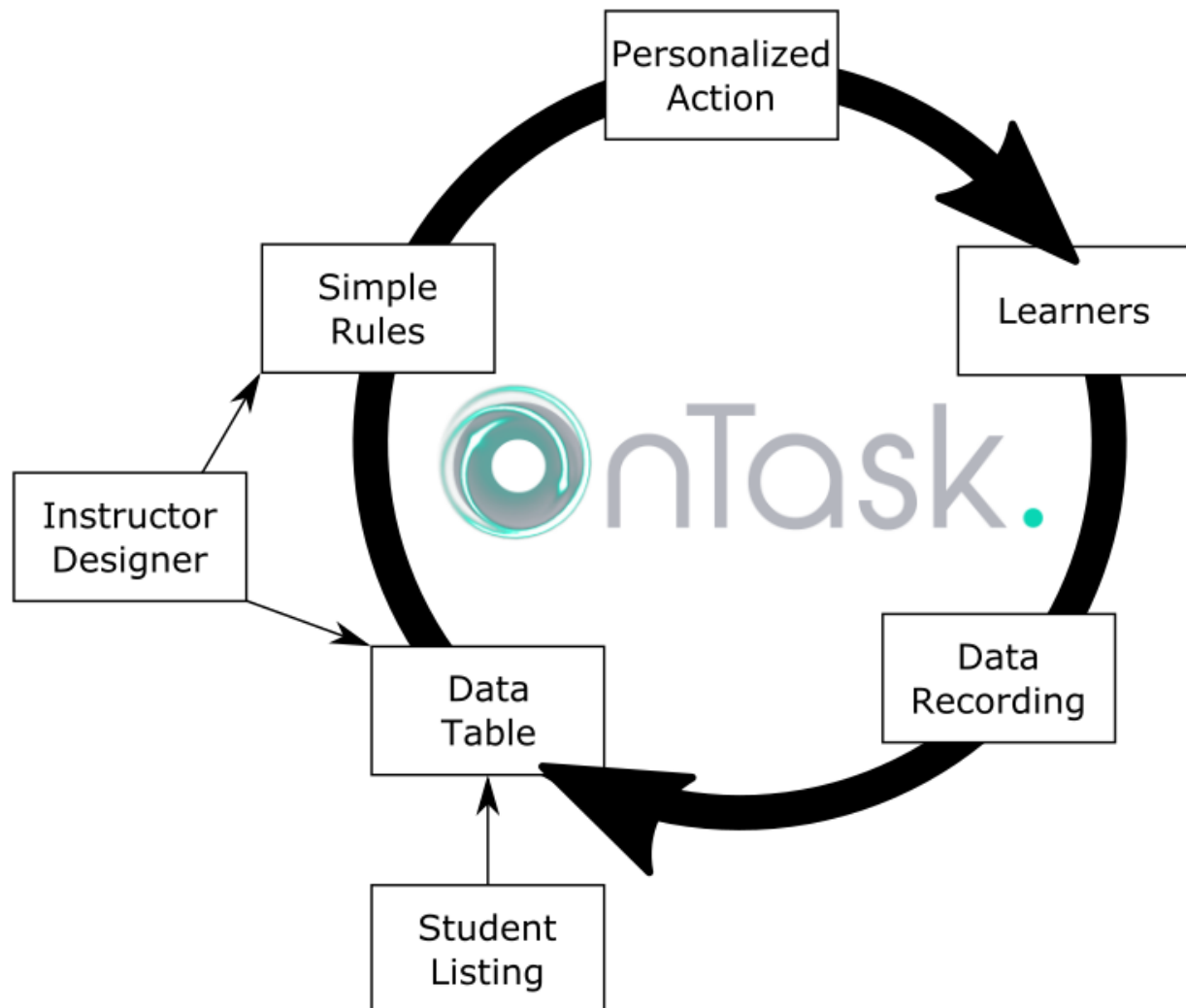
In a nutshell, the idea behind OnTask is to help instructors, learners and designers to exchange data about what is happening in a learning experience to design and deploy personalized learner support actions. This last term, *personalized support actions* is purposefully vague to include any action that is offered to learners in different forms depending on personalized conditions. The typical *workflow* to use OnTask starts by uploading and combining available data about the learning experience, either extracted from some platform (LMS, video annotation, quizzes), provided by the learners (questionnaires), or captured by the instructors. The instructors then write simple rules to select a subset of students and create a resource (HTML page, a message, a visualization) so that its content depends on the data available for each learner. The following figure shows an illustration of this workflow:

Imagine a learning experience in which you want to provide three personalized messages to the learners. In the first week, you want to send a welcome email and personalize slightly the text of this message based on the student background (courses taken before this one). The second week you want to send some comments and suggestions about the participation in the forum and the text will depend on the measures of engagement obtained from the platform. The third personalized email will vary depending on the level of engagement with the videos in the course. The idea of these messages is that you want to change the text in the message for each learner based on the information stored in the table.

The main entity in OnTask is a *workflow* that contains a **table** with data about the learners and a set of **actions**. You first need to populate the table with data extracted from the learning environment. We assume that there several data sources (coming from the use of technology, self-reported by the students, or observed and reported by the instructors). These data sources are combined and stored in the **table**. Think of the table as a conventional excel sheet storing the information about the learners (one learner per row and a set of features about each learner as columns).

The third entity in OnTask is the **action** that is either a set of questions to collect information from the learner or a text with elements that are selected or ignored based on **conditions** that are evaluated with respect to the learner student features stored in the table. This text can be included in an email, made available through a web page, or forwarded to another system for further processing.

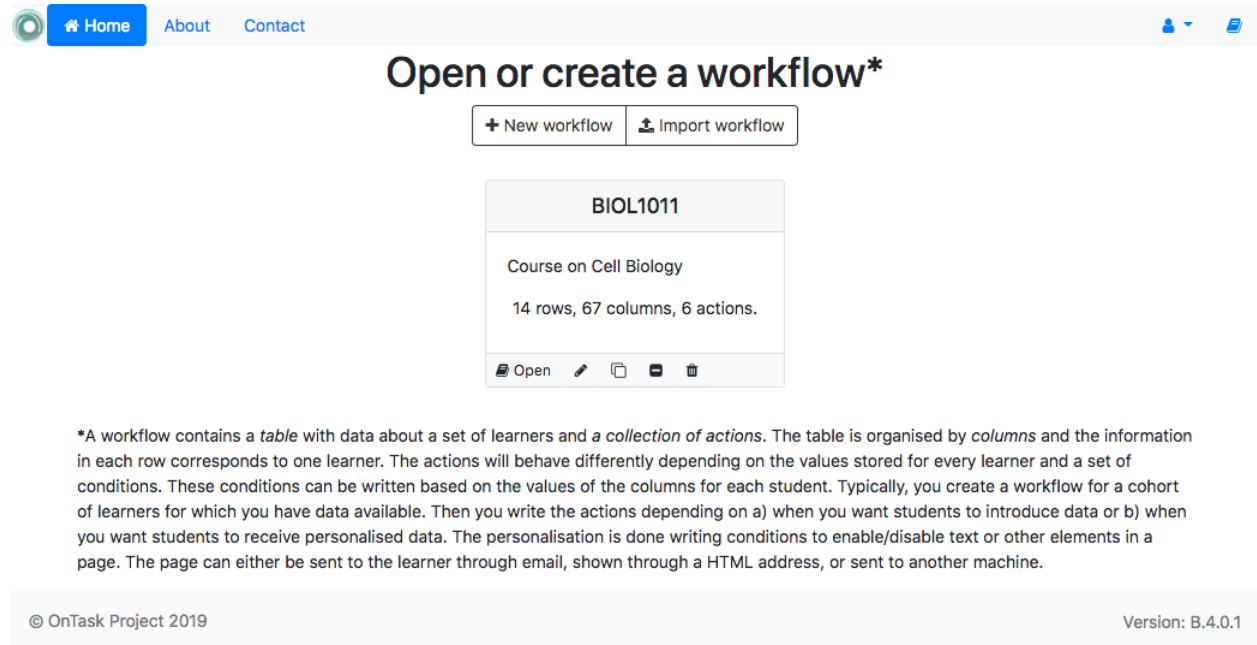
In the remainder of this section we will use a `initial workflow` that can be downloaded and imported to your collection of workflows in OnTask.



1.3.1 Workflow

“But anyone who has experienced flow knows that the deep enjoyment it provides requires an equal degree of disciplined concentration.” — Mihaly Csikszentmihalyi

After logging into OnTask, the users sees the home page showing the collection of workflows available to the user as shown in the following figure.



The navigation through the platform is done using the links in the top menu bar. The *Home* icon in the left side of the menu brings the user back to the page showing the available workflows. The book icon in the upper right side of the menu bar opens the documentation page, and the user icon provides access to the user profile (name, last name, bio, password and authentication tokens).

The **Home** page offers the following operations:

Going back

The **Home** link at the left corner of the top menu *closes* the workflow being manipulated and shows you the list of available workflows. The workflow is also *unlocked* so that it can be accessed by those users *sharing* its access.

1. *New workflow* opens a dialog to set the name and description of a new workflow and then show the page to upload data in the table.
2. *Import workflow* opens a dialog requesting a workflow name and a file containing a previously exported OnTask workflow as shown in the following figure:

This page also shows those workflows previously created by the user. Each workflow is shown as depicted in the following figure.

The icons in the bottom of this element allow you to perform the following operations:

1. *Open the workflow* selects the workflow for further manipulation. Once selected, workflow name is shown under the top navigation bar as illustrated in the following figure.

When a workflow is open, the top-bar menu appears with the structure shown in the following figure.

Create a workflow ✕

Create a new *empty* workflow (that is, with no data table or actions) by providing a name and a description. The next page will ask you for the data to populate the data table.

Name*

Description text

Import workflow

Name*

File*






initial_workflow.gz

File containing a previously exported workflow

BIOL1011

Course on Cell Biology

14 rows, 67 columns, 6 actions.

 Open    

The menu contains the following links:

Home Link back to the home page showing the workflows available for manipulation.

Actions Create, edit and execute actions.

Table Operations to visualize and manipulate the table (search for values, add a row, add a column)

Settings Additional operations to view:

Workflow operations This page contains links to export, rename, clone, flush the data, or delete the workflow. It also allows you to manipulate a set of attributes, and add users to share the workflow.

Column operations This page shows information about the columns in the workflow.

Scheduled actions Shows which actions have been scheduled for execution.

View logs A table showing the history of operations performed on this workflow

User profile Link to the user profile and logout

Manual Link to the OnTask manual (this document)

Once you open a workflow, the platform *locks it* so that no other user can manipulate it (see [sharing a workflow](#)). The lock is released if the user goes back to the home page, logs out, or the session expires. If you access a workflow and another user is currently using it, the platform will tell you that it is locked and show who is holding the lock.

2. *Change name or description* changes the name of the description given to the workflow.
3. *Create a duplicate* or exact copy of the workflow with the prefix *Copy_of_* added to the workflow name.
4. *Delete all data* and actions stored in the workflow (turns it into an empty workflow with just the name and the description).
5. *Delete the workflow* from the system.

1.3.2 Data

“May be stories are just data without a soul” – Brené Brown

This section describes the operations to either upload the first set of data into the table, or merge additional data with the one already stored. This step may be done automatically before you work with a workflow. If this is the case, you may skip this section or revisit it when you need to manipulate the existing data.

The data operations are divided into three categories: upload operations, merge operations, and plugin execution. All of them, together with the possibility of removing the data in the table are shown in the pull-down menu with name **Manage table data**.

If the workflow table is empty, OnTask will request the parameters required to [upload data](#) from a given data source. On the other hand, if the workflow table already contains data, OnTask will request the parameters to perform a *merge operation* in which the existing data is combined with the data extracted from the given source.

Upload Data

These operations are provided to upload the initial set of data into the workflow table using a variety of sources.

Upload CSV Files

CSV or “comma separated value” files are plain text files in which the first line contains a comma-separated list of column names, and every subsequent line contains the values of these columns for each row. It is a popular format to exchange data that can be represented as a table (Microsoft Excel allows to save one sheet in a spreadsheet file into this format). The following figure shows the first step to perform this operation.

BIOL1011

Step 1: Select CSV file

initial_workflow.csv
File in CSV format (typically produced by a statistics package or Excel)

Lines to skip at the top

Number of lines to skip at the top when reading the file

Lines to skip at the bottom

Number of lines to skip at the bottom when reading the file

© OnTask Project 2019 Version: B.4.0.1

In some cases, the file with comma-separated values contains several lines at either the top or the bottom of that need to be skipped when processing the data. The page to upload the CSV file allows you to specify the number of lines to skip at the start and end of the file.

Upload Excel Files

OnTask is also capable of uploading the data from one sheet of a Excel file. The following figure shows the first step to perform this operation.

BIOL1011

Step 1: Select Excel file

No file selected.
File in Excel format (.xls or .xlsx)

Sheet*

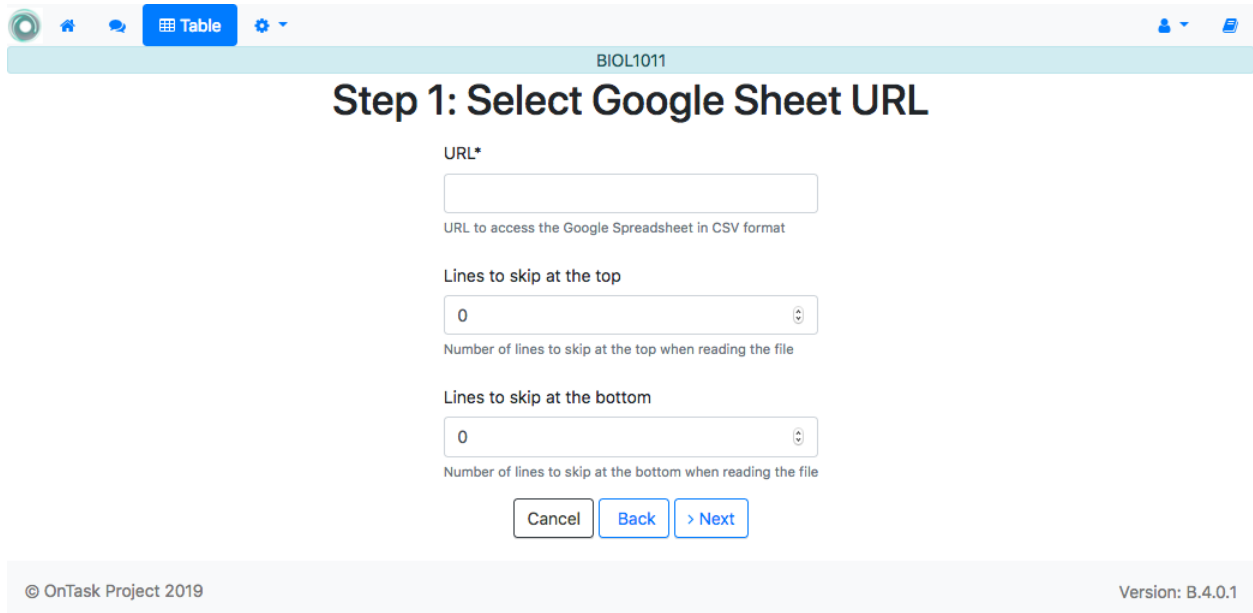
Sheet within the excel sheet to upload

© OnTask Project 2019 Version: B.4.0.1

In this case the file is assumed to have multiple *Sheets* and one of them has to be selected to upload the data.

Upload Google Spreadsheet Files

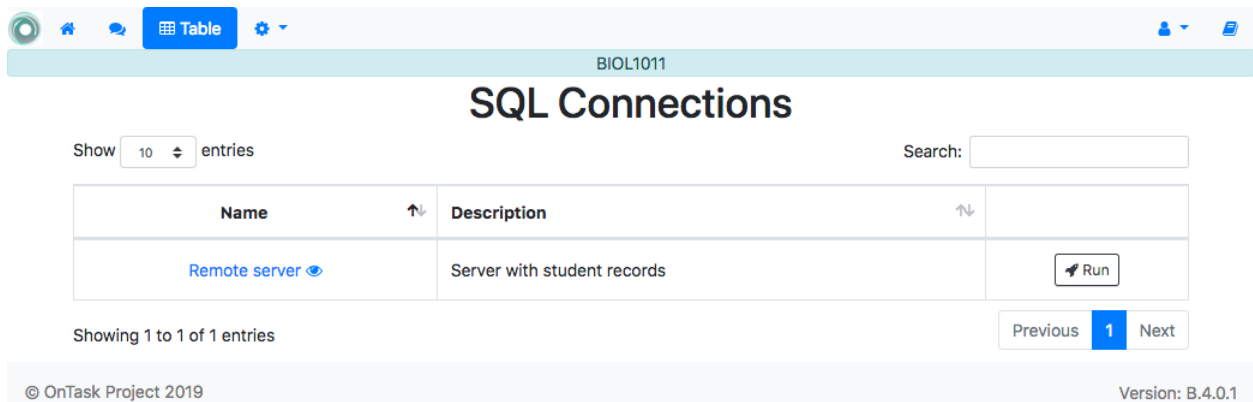
OnTask allows you to upload a data table stored in a Google Spreadsheet that is publicly accessible.



The screenshot shows the OnTask web interface. At the top, there's a navigation bar with a 'Table' button and a user profile icon. Below the navigation bar, the page title 'BIOL1011' is displayed. The main heading is 'Step 1: Select Google Sheet URL'. The form contains three input fields: 'URL*' (with a placeholder for the Google Spreadsheet URL), 'Lines to skip at the top' (set to 0), and 'Lines to skip at the bottom' (set to 0). Each input field has a small description below it. At the bottom of the form, there are three buttons: 'Cancel', 'Back', and '> Next'. The footer shows '© OnTask Project 2019' and 'Version: B.4.0.1'.

SQL connection

This operation uploads the data into the current workflow table using a SQL connection to a remote database. These connections have to be *previously defined and configured by the system administrator*. Instructors can use them to access the content of a previously defined table in a remote database. Once selected, the platform shows the SQL connections available and the possibility to view the connection parameters (click on the connection name), or *Run* the connection to upload the data as shown in the following figure.



The screenshot shows the OnTask web interface with the 'SQL Connections' table. The table has two columns: 'Name' and 'Description'. There is one entry: 'Remote server' with a description 'Server with student records'. To the right of the entry is a 'Run' button. The table is paginated, showing '1 of 1 entries'. The footer shows '© OnTask Project 2019' and 'Version: B.4.0.1'.

Name	Description
Remote server	Server with student records

When *running* a SQL connection the platform shows the configuration parameters and requests the password to access the remote database (if required).

BIOL1011

Step 1: Establish a SQL connection

The data will be accessed with the following parameters:

Connection type: mysql

Username: remote_db_user

Host: dbserver.bogus.com

Database name: demographics

Database table: s_records

Cancel Back > Next

© OnTask Project 2019 Version: B.4.0.1

Confirmation step to upload data

When uploading data for the first time, the values are prepared to be assigned as the initial content of the workflow table. But before this assignment is done, the platform needs you to verify some information. Upon reading the new data, OnTask will automatically detect the data type in each column and those columns that have unique values (no repetitions) and mark them as **keys**. Key columns are very important because their values (as they are different for every row) are required for several operations. The workflow table **must have at least one key column**. If here are several columns with this property, OnTask *allows you to *unmark* some of them as non-key* as long as there is always one of them with such mark. Additionally, you may *mark any column as a key column* if the values are all different. The operations to manipulate column information is described in the section *Workflow Operations*.

Before finishing the upload step and storing the data in the table, OnTask also allows you to change the name of the columns or change the **Key** attribute as shown in the following figure.







After this step the data is stored and the platform shows the *Table* page.

Data Merge

Merge a.k.a “Join”

Merging is a common operation in databases and is commonly known as *join*. There are several variants of join operations depending how the differences between the key columns are handled. These same variants exist in OnTask when combining the data already existing in the table. The operation relies on the **key column** to merge the two sources.

A merge operation is executed when data is uploading and the workflow **already has data in its table**. Although this operation is common in data science contexts, it has several variants that make it challenging to use properly. These variants derive mostly from the method used to specify how the values in the new columns are *matched* with respect to the ones already existing in the table. In other words, each new column has a set of values, but they need to be in the right order so that the information is matched appropriately for every row. For example, if the table contains a column with the age of the learners, and a new column with the gender is merged, the rows of the new column need to correspond with the learners in the existing table. The way to address this issue is to use a **key column* in the existing table and another ** key column** in the new data. These columns uniquely distinguish each row with a value so they are used to make sure that the information for the rows with matching values in these columns are merged. These operations are executed in a set of additional steps. The first step of the merge operation is identical to




Table

BIOL1011



Step 2: Select Columns

Cancel
Back
> Next

Load <input checked="" type="checkbox"/>	Name	Rename to	Type	Key
<input checked="" type="checkbox"/>	SID	<input type="text" value="SID"/>	number	Keep? <input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	email	<input type="text" value="email"/>	string	Keep? <input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Surname	<input type="text" value="Surname"/>	string	Keep? <input type="checkbox"/>
<input checked="" type="checkbox"/>	GivenName	<input type="text" value="GivenName"/>	string	Keep? <input type="checkbox"/>
<input checked="" type="checkbox"/>	MiddleInitial	<input type="text" value="MiddleInitial"/>	string	
<input checked="" type="checkbox"/>	Gender	<input type="text" value="Gender"/>	string	
<input checked="" type="checkbox"/>	UOS Code	<input type="text" value="UOS Code"/>	string	
<input checked="" type="checkbox"/>	Program	<input type="text" value="Program"/>	string	
<input checked="" type="checkbox"/>	Enrolment Type	<input type="text" value="Enrolment Type"/>	string	
<input checked="" type="checkbox"/>	Attendance	<input type="text" value="Attendance"/>	string	
<input checked="" type="checkbox"/>	Difficult Week 1	<input type="text" value="Difficult Week 1"/>	number	
<input checked="" type="checkbox"/>	difficult_week02	<input type="text" value="difficult_week02"/>	number	
<input checked="" type="checkbox"/>	Q01	<input type="text" value="Q01"/>	number	
<input checked="" type="checkbox"/>	Q02	<input type="text" value="Q02"/>	number	
<input checked="" type="checkbox"/>	Q03	<input type="text" value="Q03"/>	number	

the upload operation. After detecting the column data types, the key columns and offering the option of changing their names, the next steps identify the key columns to use in the merge, the variant to merge, and shows a summary of the changes that will result from the operation.

Step three: select keys and merge option

The following figure shows the third step of the merge operation.

Step 3: Select Keys and Merge Option

Cancel Back > Next

Key Column in Existing Table*

SID

Key column in the existing table to match with the new table.

Key Column in New Table*

SID

Key column in the new table to match with the existing table.

Method to select rows to merge/update*

3) Select only the rows with keys in the existing table

Select one method to see detailed information

Selects only the rows with key values present in the existing table. This method drops those rows in the new table with key values not in the existing one (rows 5 and 6 in the figure below). The cells in the result rows are updated only if the new value is non-empty (cell 4B remains with value b).

Update/Merge

Existing			New		
K	A	B	K	B	C
✓ 1	a	b			
✓ 2	a	b			
✓ 3	a	b	3	b'	c
✓ 4	a	b	4		c
			5	b'	c
			6		c

Existing

Result

K	A	B	C
1	a	b	
2	a	b	
3	a	b'	c
4	a	b	c

Cancel Back > Next

© OnTask Project 2019

Version: B.4.0.1

The form requires the following fields:

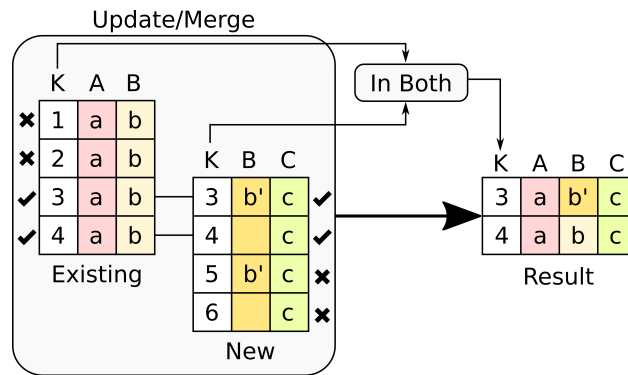
Key columns A key column in the external table about to be merged and a key column in the existing table (both fields are required).

Merge method After choosing a merge method, a figure and explanation are shown below.

There are four possible merge variants:

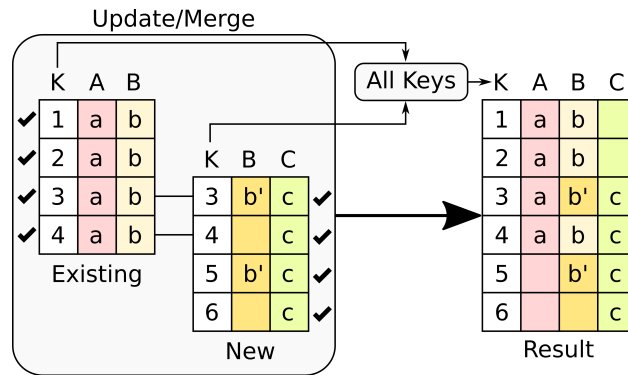
- 1) **Select only the rows with keys in both existing **and new table**.** It will select only the rows for which values in both key columns are present. Or in other words, any row for which there is no value in either of the

key columns **will be dropped**.



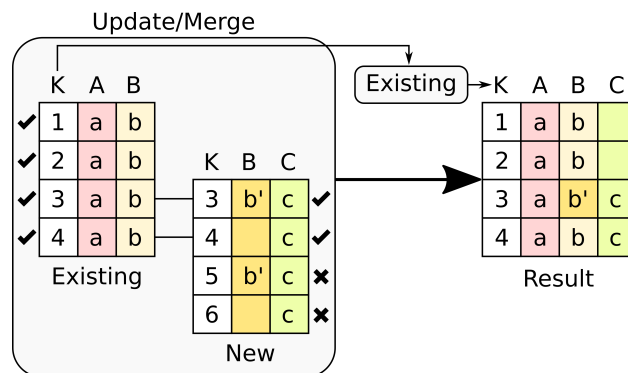
2) Select all rows in either the existing or new table.

All rows in both tables will be considered. You have to be careful with this option because it may produce columns that are no longer unique as a result.



3) Select the rows with keys in the existing table.

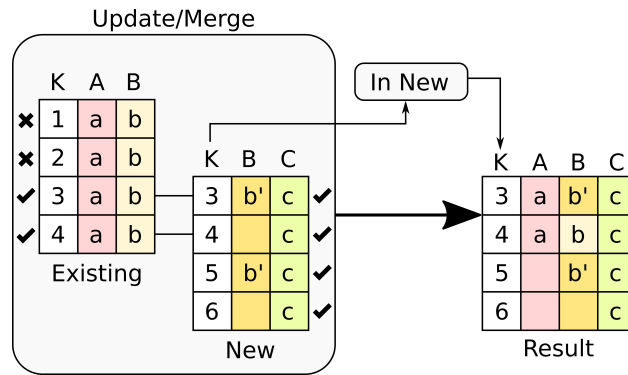
Only the rows in the new table with a value in the key column that is present in the existing table will be considered, the rest will be dropped.



4) Select the rows with keys in the new table.

Only the rows in the existing table with a value in the key column that is present in the key column from the new table will be considered, the rest will be dropped.

In any of these variants, for those columns that are present in both the existing table and the new table, the values of the second will update the existing ones in the first. This update may introduce non-values in some of the rows (for



example in columns for with the new data does not provide any value). Additionally, extra care needs to be taken when performing this operation as some of the merge variants may eliminate data in the existing table. In the extreme case, if you try to merge a table with a key column with no values in common with the existing key and you select the method that considers rows with keys in both the existing and new table, the result is an empty table.

Step four: verify upcoming changes

After selecting these parameters the last step is to review the effect of the operation and proceed with the merge as shown in the following figure.

1.3.3 Table

“You’re here because you know something. What you know you can’t explain, but you feel it” – Morpheus, The Matrix

The information stored in the workflow table is accessed through the *Table* link in the top bar menu. These pages show the values stored in the workflow table. As this table can be arbitrarily large, OnTasks groups the rows into pages and offers the possibility to choose how many rows are contained in each page, as well as links to each of the pages. If the table has a large number of columns, a horizontal scroll is available to show the additional content. The order of the columns in the table can be changed by dragging them from the top row and dropping them in a new position.

The following figure shows an example of the main table page.

The table also offers the possibility of searching for a value. The given string is used to search all the columns in the table.

The buttons at the top of the page offer the following functions:




Row A form to introduce a new row in the table with one field per column.

Column Add a new column to the table. The pull-down menu also allows to add a column with values calculated with a formula and using the existing values, or a column with random values (See [Adding a column](#) for a detailed explanation).


Manage table data Menu to upload/merge data to the table (see [Data](#) for a detailed explanation), execute a plugin, or delete the data in the table.

Views A view is a screen showing a subset of rows and/or columns in the table. Clicking in the Views link will take you to another screen to [manage these views](#). The pull down menu offers links to access the subset of the table defined by the available views.



Dashboard The dashboard shows a statistical summary for the columns shown in the table. This number may be too high, so you should consider using [Table Views](#) to reduce the amount of information shown in the dashboard.

Table









BIOL1011

Step 4: Review and confirm

Existing Table		New Data
SID (Key)	←	SID (Key)
Attendance (Update)	←	Attendance
Contributions (Update)	←	Contributions
Contributions 2 (Update)	←	Contributions 2
Contributions 3 (Update)	←	Contributions 3
Contributions 4 (Update)	←	Contributions 4
Contributions 5 (Update)	←	Contributions 5
Correct_1_W2 (Update)	←	Correct_1_W2
Correct_1_W3 (Update)	←	Correct_1_W3
Correct_1_W4 (Update)	←	Correct_1_W4
Correct_1_W5 (Update)	←	Correct_1_W5
Correct_2_W2 (Update)	←	Correct_2_W2
Correct_2_W3 (Update)	←	Correct_2_W3
Correct_2_W4 (Update)	←	Correct_2_W4
Correct_2_W5 (Update)	←	Correct_2_W5
Days online (Update)	←	Days online
Days online 2 (Update)	←	Days online 2
Days online 3 (Update)	←	Days online 3




Table


































BIOL1011

Table

+ Row
+ Column ▾
Manage table data ▾

Views ▾
Dashboard
CSV Download

Show 10 entries Search:

	SID ↑↓	email ↑↓	Surname ↑↓	GivenName ↑↓	MiddleInitial ↑↓	Gender ↑↓	UOS Code ↑↓	P
  	300773633	euhz6532@bogus.com	Jen	Jie	M	male	ELON3509	
  	315500979	zqdk2609@bogus.com	Iredale	Flynn	M	male	ELON3509	
  	317547296	ckrn7263@bogus.com	Prenzel	Blake	I	male	ELON3509	
  	319549896	ttqp9766@bogus.com	Chien	Jin	Y	male	ELON3509	
  	320250819	tjxk6150@bogus.com	Tuan	Donglu	D	female	ELON3509	
  	326285587	isim6886@bogus.com	Pan	Li Na	Y	female	ELON3509	
  	330794979	olie8242@bogus.com	Chang	Lian	M	female	ELON3509	
  	352120481	ucdx8510@bogus.com	Shao	Tain	L	male	ELON3509	
  	360594548	euho9752@bogus.com	Vida	Bailey	C	male	ELON3509	
  	363393232	tdrv2640@bogus.com	Hague	Joel	K	male	ELON3509	

Showing 1 to 10 of 14 entries Previous **1** 2 Next

You may drag-and-drop a column header to change its position in the table

© OnTask Project 2019
Version: B.4.0.1

+ Row
+ Column ▾
Manage table data ▾

Views ▾
Dashboard
CSV Download

CSV Download This functionality downloads the data in the table as a CSV file. Combine this functionality with the *Table Views* to handle large tables.

The left-most column in the table shows the operations to manipulate a row:

Edit content This link opens a form with one field per column to modify the values in the row. The key columns are shown but not allowed to be changed.

Dashboard A dashboard showing the population measures for all the columns and a mark showing where are the values for this row.

Delete Delete the row from the table.

Table Views

Due to the potentially large size of the workflow table in either number of rows or columns, OnTask offers the possibility to define *views*. A view is a table that shows a subset of columns and rows of the original table. You may define as many views as needed. The link *Views* in the main table page shows the views available and the operations to manage them.

The screenshot shows the 'Table Views' interface for a course named 'BIOL1011'. At the top, there are navigation icons and a 'Table' button. Below the course name, the title 'Table Views' is displayed, along with 'Full Table' and '+ Add View' buttons. A search bar is located to the right of the table. The table itself has two main columns: 'Name' and 'Description'. It lists three views: 'Low forum sessions', 'Midterm', and 'Videos'. Each view entry includes a set of icons (eye, copy, delete) for management. Below the table, it indicates 'Showing 1 to 3 of 3 entries' and provides 'Previous', '1', and 'Next' navigation options. The footer contains the copyright notice '© OnTask Project 2019' and the version 'Version: B.4.0.1'.

The buttons at the top of the page offer the following operations:

Full Table Go back to the page showing the entire data table.

Add View Create a new view

When creating a view you need to provide a name (required), a description, a subset of columns to show (at least one of them must be a **key column**), and an expression to select a subset of rows. This expression is evaluated with the values of every row and if the result is *True*, the row is included in the view. The following figure shows an example of the information that is included in the definition of a view.

Once a view is created, the following operations are available:

Edit Click in the view name to edit its elements (name, description, set of columns to show and row filter)

Show Show the subset of the table selected by the view

Clone Create a duplicate of this view with the prefix *Copy_of* added to its name. This operation is useful to create a new view with content that is similar to an already existing one (clone and edit).

Edit view ✕

View name*

Midterm

View Description

Columns to show

☐ email
 ☐ Q01
 ☐ Q02
 ☐ Q03
 ☐ Q04
 ☐ Q05
 ☐ Q06
 ☐ Q07
 ☐ Q08
 ☐ Q09
 ☐ Q10
 ☐ Total

Click here to search ▼

Row will be shown if

Delete Delete the view.

The page to show the subset of the table defined by the view is identical to the full table view, but the table is restricted to the appropriate columns and rows. The view shown in the previous figure defines a subset of 12 columns of the table that is rendered as shown in the following figure.

The *Dashboard* and *CSV Download* buttons, when used while in a view, will apply to the selected data subset.

1.3.4 Actions

“In order to carry a positive action we must develop here a positive vision” – Dalai Lama

This is the most important functionality of the platform. Actions are used exchange information with the learners or other platforms, either through personalized information, or requesting data through a survey. A workflow contains a set of actions shown when selecting *Actions* page in the top-bar menu. The next figure shows an example of the actions available in one workflow.





The buttons at the top of the page offer the following operations:

Action Create a new action in the workflow. The form requires a name (unique for the current workflow), a description (optional), and the type of action. OnTask offers the following types of actions: personalized text, personalized Canvas email, personalized JSON, and surveys.

Import action Upload an action previously downloaded from another workflow.

The actions in the workflow are shown in a tabular format. For each action the following main operations are offered:

Edit Click in the name action to edit its content.




Table
































BIOL1011

Midterm

+ Row
+ Column
Manage table data

Views
Edit view
Full table
Dashboard
CSV Download

Show 10 entries Search:

	email	Q01	Q02	Q03	Q04	Q05	Q06	Q07	Q08	Q09	Q10
  	ckrn7263@bogus.com	1	0	0	1	0	1	1	1	1	
  	ctfh9946@bogus.com	0	0	0	0	0	0	0	0	0	
  	euho9752@bogus.com	1	1	1	1	0	0	1	1	0	
  	euhz6532@bogus.com	1	0	1	1	1	1	1	1	0	
  	isim6886@bogus.com	1	1	1	0	1	1	1	1	1	
  	kjbc6748@bogus.com	1	1	1	1	0	1	1	0	1	
  	olie8242@bogus.com										
  	pzaz8370@bogus.com	1	1	0	0	1	1	0	0	1	
  	tcnf7608@bogus.com	1	0	0	0	1	1	0	1	0	
  	tdrv2640@bogus.com	0	0	1	0	1	1	1	1	1	

Showing 1 to 10 of 14 entries

Previous
1
2
Next

You may drag-and-drop a column header to change its position in the table

© OnTask Project 2019
 Version: B.4.0.1

BIOL1011

Actions

+ Action Import action Manage table data ▼

Show 10 entries Search:

Type ↑↓	Name ↑↓	Description ↑↓	Last executed ↑↓	
	Badges	A badge reflecting the activity with respect to the videos	—	Run ...
	Comments about how to prepare the lecture (Week 4)	A message commenting on the videos they watched for Week 4	—	Run ...
	Midterm comments	Comment the midterm scores divided by topic	—	Run ...
	Send JSON to remote server	Send a JSON object to a remote server (outside this platform)	—	Run ...
	Student comments Week 1	Feedback about today's lecture	—	Run ...
	Suggestions about the forum	Message to remind those students that haven't visited the forum to do so.	—	Run ...

Showing 1 to 6 of 6 entries

Previous 1 Next

© OnTask Project 2019 Version: B.4.0.1

Run Use the action to either provide personalized content or run a survey (see [Running actions](#) for more information)

URL Provide access to learners to the content of the action through a link (only available for actions of type Personalized Text)

ZIP Download a ZIP file with as many files as selected learners in the action. Each file contains the personalized document for the learner (only available for Personalized Text actions)

Schedule Schedule the execution of the action for some time in the future

Rename Edit the name and description of the action.

Clone Create an exact duplicate of the action adding the prefix “Copy_of” to its name.

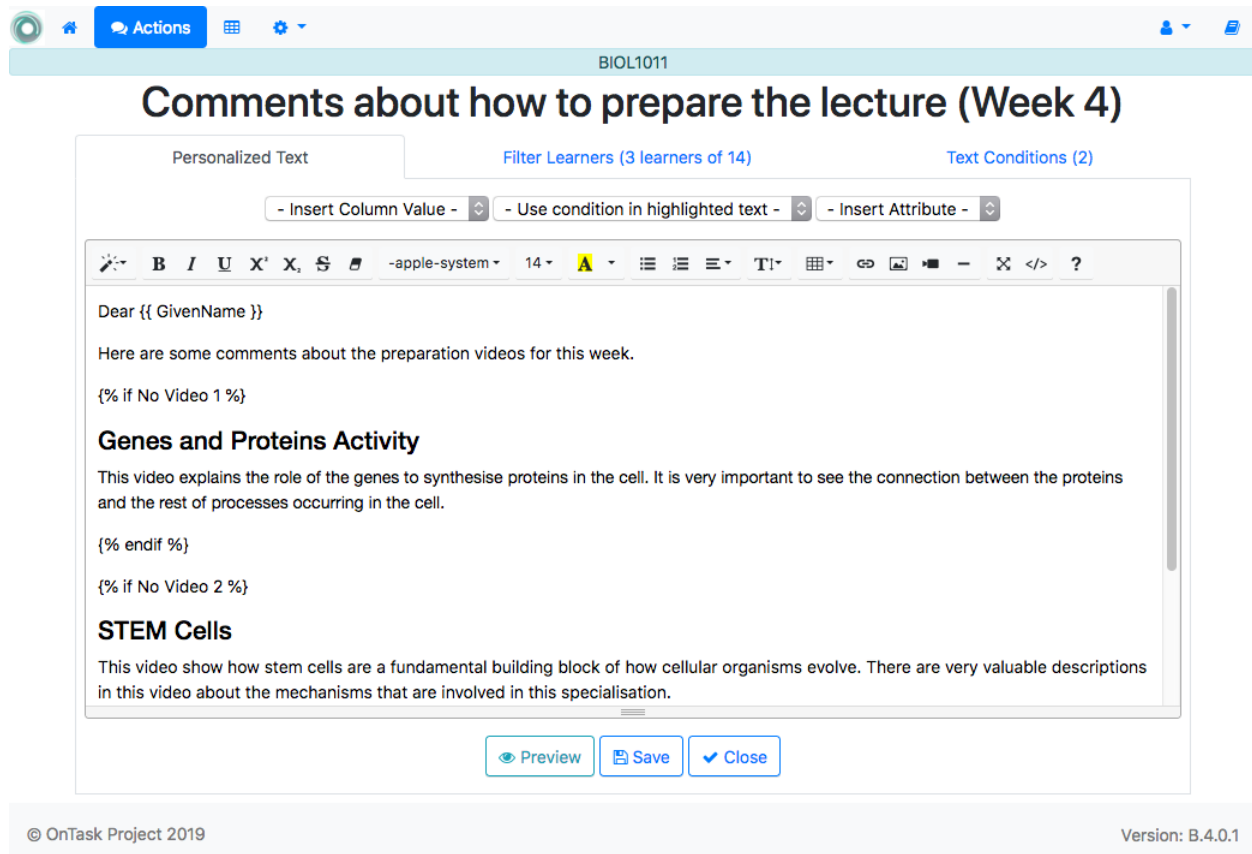
Export Download a file containing the definition of the action suitable to be uploaded into another workflow.

Delete Remove the action from the workflow.

Personalized Text

These actions allow to create a document (similar to a HTML page) and mark elements (paragraphs, sentences, images) with *conditions* that will control if they are included or ignored when showing the document. The conditions are stated in terms of the columns of the data table. Think of this personalized content as a resource (message, tip, comment) you would offer learners but with content that is different depending on the data stored in the table. You may have several of these actions prepared to be used at different points during a learning experience. The personalized text action is manipulated with the screen shown in the following figure:

The screen has three tabs: the left one contains the editor, the center one the definition of a filter (optional) to select a subset of the learners to consider for this action, and the right tab contains the conditions used in the text (if any).



The Personalized Text HTML Editor (left tab) This is a conventional HTML editor offering the usual operations (inserting text, headings, lists, links, images, etc.) Right above the editor window you have three pull down menus to: insert a column value in the text (a placeholder), use a condition to conditionally show the text currently highlighted, or insert a *workflow attribute* that will be replaced by the corresponding value.

The conditions The right tab contains the *conditions*. A condition is an expression that when evaluated with respect to the values in the table for each learner will either be **True** or **False**. These expressions are commonly used in other applications such as spreadsheets or programming languages. The following screen shows an example of the content of this tab with two conditions.

The button Condition at the top of the tab opens the form to define a new condition. Once created The buttons in the screen allow you to edit the expression, insert the condition to control the appearance of text in the editor (below), clone the condition, or delete it from the action.

Each condition shows the number of learners for which the expression in that condition evaluates to **True** (if this value is zero, it means that any text you include in the editor controlled by this condition will not appear for any of the learners), the name, description, and the defined formula.

The icons in the bottom of the condition element allow the following operations:

Edit Open a form to edit the name, description and expression in a condition.

Clone Create an exact duplicate of this condition with the prefix “Copy_of” added to its name. This operation is useful when creating a new condition with an expression very similar to an existing one.

Delete Delete the condition from this action.

The following image shows an example of this condition.

The expression in the previous condition is shown under the title **The text will be shown if** and can be read as:

BIOL1011

Comments about how to prepare the lecture (Week 4)

Personalized Text Filter Learners (3 learners of 14) Text Conditions (2)

- Insert Column Value - - Use condition in highlighted text - - Insert Attribute -

Dear {{ GivenName }}

Here are some comments about the preparation videos for this week.

{% if No Video 1 %}

Genes and Proteins Activity

This video explains the role of the genes to synthesise proteins in the cell. It is very important to see the connection between the proteins and the rest of processes occurring in the cell.

{% endif %}

{% if No Video 2 %}

STEM Cells

This video show how stem cells are a fundamental building block of how cellular organisms evolve. There are very valuable descriptions in this video about the mechanisms that are involved in this specialisation.

Preview Save Close

© OnTask Project 2019 Version: B.4.0.1

BIOL1011

Comments about how to prepare the lecture (Week 4)

Personalized Text Filter Learners (3 learners of 14) Text Conditions (2)

+ Condition

No Video 1	No Video 2
(1 learner)	(3 learners)
Didn't watch Video 1 Week 4	Didn't watch Video 2 Week 4
Formula Correct_1_W4 equal to 0.0	Formula Video_2_W4 equal to 0.0
✎ 📄 🗑	✎ 📄 🗑

Preview Save Close

© OnTask Project 2019 Version: B.4.0.1

Correct_1_W4 equal to zero

The first element is the column name `Correct_1_W4`, followed by the equal sign, and then the constant zero. When evaluating this expression, the column name is replaced by the value from the corresponding to each learner. For example, if a given learner has `Correct_1_W4` equal to 3 the evaluation will transform the expression into $3 = 0$ which is **False**. Another possible evaluation is if another learner has `Correct_1_W4` equal to zero. After substitution of the column by the values for the learner, the resulting expression is $0 = 0$. In this case, the expression is **True**. Once defined this condition can be applied to a part of the personalized text. When creating the texts, the condition is evaluated with the values for each student. If the expression is true, the text is included, if not, it is ignored. This mechanism is at the heart of how OnTask personalizes the content of the actions. In the example above, the expression in the condition one of the learners in the data table.

The filter The center tab shows a *filter*. This element is an expression used to decide which learners (or more precisely, the corresponding rows in the data table that) will be selected and used in this action.

The filter element shows the name, description, and the formula defined. The icons at the bottom of the object provide access to the following operations:

Edit Edit the name, description, and formula of the filter.

Delete Remove the filter from the action.

When editing or creating a filter, the form shows the information as in the following figure:

The expression in this condition is shown under the title **The text will be shown if** and can be read as:

$\text{Video_1_W4} = 0$ or $\text{Video_2_W4} = 0$

The first element of the expression is the sub-expression $\text{Video_1_W4} = 0$ which contains the variable `Video_1_W4`, the equal sign, and the constant zero. The second element is a sub-expression with the variable `Video_2_W4`, the equal sign, and the constant 0. These two sub-expressions are connected through the **OR** operator, which means that the expression will be **True** if either of the sub-expressions are **True**, and **False** in any other case. When evaluating this expression, the variables are replaced by concrete values (numbers). For example, if `Video_1_W4` is replaced by 3, and `Video_2_W4` is replaced by 4, the evaluation will transform

The screenshot shows the OnTask interface with a header bar containing 'BIOL1011'. The main title is 'Comments about how to prepare the lecture (Week 4)'. Below the title, there are three tabs: 'Personalized Text', 'Filter Learners (3 learners of 14)', and 'Text Conditions (2)'. The 'Filter Learners' tab is active, displaying a 'Filter' box. The filter box contains the text: 'Either Video 1 or Video 2 in Week 4 have zero engagement' and the formula: '(Video_1_W4 equal to 0.0) OR (Video_2_W4 equal to 0.0)'. Below the filter box are three buttons: 'Preview', 'Save', and 'Close'. At the bottom of the interface, there is a footer with '© OnTask Project 2019' on the left and 'Version: B.4.0.1' on the right.

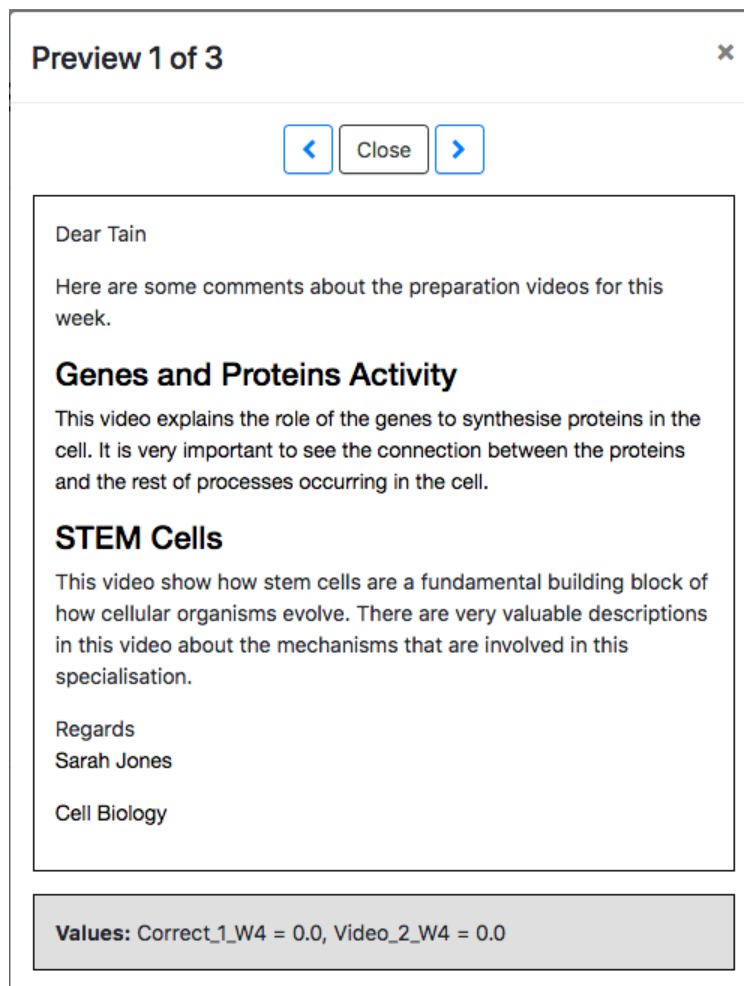
The 'Edit filter' dialog box has a title bar with a close button (X). The 'Description' field contains the text: 'Either Video 1 or Video 2 in Week 4 have zero engagement'. Below the description, the text 'The learner will be selected if' is followed by a rule editor. The rule editor has three tabs: 'NOT', 'AND', and 'OR', with 'OR' selected. There are two buttons: '+ Add rule' and 'Add group'. The rule editor contains two rules. The first rule has a dropdown menu with 'Video_1_W4', a dropdown menu with 'equal', and a text input field with '0'. There is a 'Delete' button (X) next to the rule. The second rule has a dropdown menu with 'Video_2_W4', a dropdown menu with 'equal', and a text input field with '0'. There is a 'Delete' button (X) next to the rule. At the bottom of the dialog box, there are two buttons: 'Cancel' and 'Update'.

the expression into $3 = 0 \text{ or } 4 = 0$. The sub-expression $3 = 0$ is clearly **False** and so is the other sub-expression $4 = 0$. This means the initial expression is **False**. result is either **True** or **False**. Another possible evaluation is if `Video_1_W4` is equal to zero (and `Video_2_W4` remains equal to 4). In this case the resulting expression is $0 = 0 \text{ or } 4 = 0$. In this case, the first sub-expression is **True**, and although the second is **False**, only one is needed for the overall expression to be **True**.

These conditions can have nested sub-expressions and get complex fairly quickly. However, the underlying mechanism to evaluate them remains the same: replace variables with values and decide the result (**True** or **False**). OnTask relies on these expressions to personalize the content of the actions.

The buttons at the bottom of the page offer the following operations:

Preview The Preview button shows how the text in the editor is shown for those learners selected by the filter (if any). After clicking in the button you will see a window with the resulting text. If there are any elements in the text that are controlled by any condition, the bottom area will show their values.



Use the arrow buttons to see all the different versions of the text depending on the values stored in the table for each learner.

Save This button saves the content of the text editor and continues in the same editor page.

Close This button saves the content of the text editor and returns to the page showing all the actions in the workflow.

Using column values, attributes and conditions in a Personalized Text

As previously described the *Personalized Text Editor* may include three per-learner personalized elements: an attribute name, a column name or a portion of text marked with a condition.

Attributes Attributes are simply synonyms that you may want to use in more than one action. For example, if you have several actions that include the name of a course, instead of including that name in all actions, you may define an *attribute* with name *course name* and value *Biology 101* and include in the actions the attribute name. OnTask will replace that attribute with its value when showing the text to the learners. If you then change the name of the course (or you export this workflow and import it to be used in another course), you only need to change the attribute and the name of the course will appear correctly in all actions (in what is called a *single point of change*).

To insert an attribute name in the text simply place the cursor in the editor where you want the value of that attribute to appear and select the attribute from the area above the editor. The name of the attribute will be inserted in the text surrounded by double curly braces, (for example `{{ course_name }}`). Only *the attributes* you previously created in the details page are available.

Column names The other element that can be personalized is a column name. For example, suppose you have a column in your table with the first name of the learners. You can use the column name to personalize the greeting in the text. To insert a column name, you follow the same steps used for the attribute but this time you select the column name from the pull-down menu. You will see that the name of the column appears in the text also surrounded by double curly braces (for example `Hi {{ GivenName }}`). The double curly braces is the way OnTask has to mark that text to be personalized or replaced by the corresponding value for each learner extracted from the data table.

Conditional text Using a condition to control if a portion of the text is shown or ignored is slightly different. You need to first highlight the text you want to appear depending on the condition in the editor. Then click in the pull down menu **Use condition in highlighted text** and select the condition to use. The text will be surrounded by two marks. For example if the condition name is `No Video 1`, the text you highlighted will appear in the editor after clicking in the *Insert in text* as:

```
{% if No Video 1 %}You need to review his week's video{% endif %}
```

This format marks the message *You need to review this week's video* to appear only for those learners for which the condition `No Video 1` evaluates to **True** with their current values in the data table. Otherwise, the text will be ignored. The following figure illustrates this process.

Surveys

The personalized text actions described in the previous section makes information available to the learners. The *survey* actions perform the operation in the opposite direction, they collect information from the learners and store it in the table. This functionality can be seen as a simplified version of other survey engines such as Google Forms, SurveyMonkey or Qualtrics. In a learning context a survey can be used by the learners to submit certain data, or by the instructor to collect annotations about learners throughout the experience. OnTask supports these two modalities.

The edition of a survey action is done through a page with three tabs. The following figure shows an example of this page.

The information collected for each question will be represented in the table by a column. The editor page allows you to use any of the existing questions to be included in a survey. The three tabs in the screen offer the following functionality.

Survey Questions The tab in the left shows the questions that are contained in the survey. The two buttons at the top of the screen allow you to either insert an existing question (the pull-down menu will show all the column names available in the table) or create a new question. When creating a new question, the following form is used:

BIOL1011

Student comments Week 1

Survey Questions Filter Learners (Empty) Survey parameters

- Insert existing question - + Create question

Show 10 entries Search:

Name	Description (shown to learners)	
Difficult Week 1	What was the most difficult topic in this week's lecture?	
Sat W1	What was the most relevant topic in this week's lecture?	

Showing 1 to 2 of 2 entries

Previous 1 Next

Preview Done

© OnTask Project 2019 Version: B.4.0.1

The field *Question name* will be used internally as the column name in the table. The *Description* field is the text shown to the learners next to the question. If the question includes a set of values allowed, the form available to the students will collect the answers using a pull-down menu with the given choices.

As with other tables in OnTask, if the number of elements (in this case questions) is too large, they will be divided into pages with a link to access each page, and the content of the questions is searchable.

Filter Learners The central tab in this screen is identical to *the filter in the personalized text action*. The tab allows to include an expression to decide if a learner is included or not in the survey. This survey has no filter defined.

Survey parameters The tab in the right of the screen shows the additional parameters to deploy the survey. More precisely the screen allows to define the text that is shown at the top of the survey (*Survey description*), the key column used to identify the users, and if the questions should be shown in different order for each user.

The *Preview* button at the bottom of the page shows the content as it will be shown to the learners.

Personalized JSON Object

This type of action allows the creation of a *JSON object* with content that is personalized with the same functionality as described in the section about *Personalized Content*. The difference is that instead of creating a text, the action creates a JSON object that will eventually be sent to another platform for further processing. This object is also a resource that is different for every student but the difference is that instead of being prepared to be visualized, it is packaged with a structure suitable to be received by another platform through a URL.

The screen to create a Personalized JSON object is shown in the following figure.

The tabs have the same functionality than in the case of *personalized text*.

Personalized JSON Object This tab contains a plain text editor to describe the structure of the object and *insert column values, attribute values or use conditions to control the presence of elements in the object*.

Filter Learners This tab allows the definition of an expression to select a subset of rows in the table for processing.

Text Conditions This tab contains the conditions that can be used within the body of the JSON object to select content (in exactly the same way as in the *personalized text*).

Add question ✕

Question name*

Description (shown to the learners)

Type of data to store in the column*

number

Question position (zero to insert last)*






0

Question active from



Question active until

Comma separated list of values allowed in this column

Cancel✓ Add question



BIOL1011




Student comments Week 1

Survey Questions

Filter Learners (Empty)

Survey parameters



Survey description (shown to learners) 

Feedback about today's lecture

Key column to match the user id

email

Shuffle questions? ☐

 Preview  Done

© OnTask Project 2019

Version: B.4.0.1

Preview 1 of 14 ×

< Close >

Feedback about today's lecture

What was the most difficult topic in this week's lecture?

What was the most relevant topic in this week's lecture?

OnTask Actions BIOL1011

Send JSON to remote server

Personalized JSON Object Filter Learners (Empty) Text Conditions (2)

- Insert Column Value - - Use condition in highlighted text - - Insert Attribute -

```
{
  "sid": {{ SID }},
  "midterm_total": {{ Total }},
  "msg":
    {% if Less than 50 in the midterm %}"Message number 1"{% endif %}
    {% if More or equal to 50 in midterm %}"Message number 2"{% endif %}
}
```

Target URL

http://127.0.0.1

Preview Save Close

© OnTask Project 2019 Version: B.4.0.1

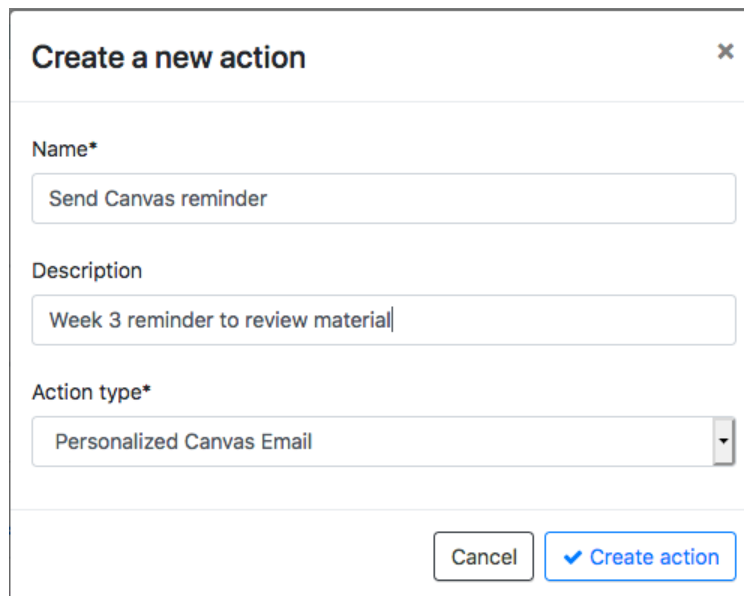
The text shown in the previous figure defines a JSON object with three fields `sid`, `midterm_total` and `msg`. The first two contain column names that will be replaced by their corresponding values. The field `msg` will include one of the two messages depending on the value of the conditions.

The field *Target URL* is to introduce the URL where the object will be sent.

The preview button in the personalized JSON action shows the resulting object after verifying that the structure after evaluating the corresponding expressions is a valid JSON object.

Personalized Canvas Email

This type of action is only available if OnTask is *appropriately configured* to communicate with a [Canvas Learning Management System](#). The creation of this type of action is almost identical to the *Personalized Text*. The action is created selecting the corresponding action type as shown in the following figure.



The page to edit this action is almost identical to the one to edit a *Personalized Text* action.

It contains three tabs: *Personalized Canvas Email*, *Text Conditions* and *Filter Learners*. The last two, *Conditions* and *Filter Learners* offer the same functionality. The *Personalized Canvas Email* allows the creation of a plain text message (no HTML markup is allowed).

Running actions

Once an action has been created, it can be *run*. The meaning of this term is different for the various types of actions supported in OnTask.

Sending personalized emails (Personalized Text Actions)

Once you created a personalized text action and verified its content using the *Preview* button, save its content. The right-most column has a button with name *Run*.

If selected, the next page is a form requesting information about how to send the messages to the learners. The next figure shows an example of this page.

The fields in this form are:

The screenshot shows the OnTask web interface for sending a Canvas reminder. The top navigation bar includes a home icon, an 'Actions' button, a grid icon, a settings icon, and a user profile icon. The course identifier 'BIOL1011' is displayed in the header. The main title is 'Send Canvas reminder'. Below the title, there are three tabs: 'Personalized Canvas Email' (selected), 'Filter Learners (2 learners of 14)', and 'Text Conditions (1)'. The email content area contains a text editor with the following text: 'Dear {{ GivenName }}', 'We recommend that you review the discussions in the online forum about the topics we are going to cover this week', 'Regards', 'John Doe', and 'Course Coordinator'. Above the text editor are three dropdown menus: '- Insert Column Value -', '- Use condition in highlighted text -', and '- Insert Attribute -'. Below the text editor are three buttons: 'Preview', 'Save', and 'Close'. The footer shows '© OnTask Project 2019' on the left and 'Version: B.4.0.1' on the right.

	Midterm comments	Comment the midterm scores divided by topic	—	Run ...
--	------------------	---	---	---------

BIOL1011

Send emails for action "Midterm comments"

14 Emails will be sent.

Email subject*

Column to use for target email address*

email

Comma separated list of CC emails

Comma separated list of BCC emails

☐ Check/exclude email addresses before sending?

☐ Send you a summary message?

☐ Track email reading in an extra column?

☐ Download a snapshot of the workflow?

A zip file useful to review the emails sent.

Cancel Preview > Next

© OnTask Project 2019 Version: B.4.0.1

Email subject A line to be included as subject of all the emails.

Column to use for target email address OnTask needs to know where to send the email. It assumes that you have a column containing that information for each learner and it needs you to select that column.

Comma separated list of CC emails A comma-separated list of emails to include in the *carbon copy* or *CC* email field.

Comma separated list of BCC emails A comma-separated list of emails to include in the *blind carbon copy* or *BCC* email field.

Check/exclude emails If selected, this option inserts an extra step in which you can eliminate certain emails from the action. This feature is useful to remove certain emails that cannot be removed with the filter.

Send you a summary message If you select this option OnTask will send you an email with the summary of this operation (number of rows in the table that were selected by the filter, number of emails sent, date/time of the operation, etc).

Track email reading Include in the messages a HTML snippet to detect if the email is read. OnTask adds an extra column to the table to store the number of times the message is opened. This detection relies on how the email client opens the message and processes the included images, therefore, the information in this column may not accurately reflect this information.





Snapshot of the workflow If you select this option, after the emails are sent, the platform returns you a file that contains a snapshot (picture) of the workflow. It basically freezes the content of the workflow and places it in a file given to you. You may take this file and *import back the workflow*. In this new workflow you can check the values and messages at the time the operation was executed.

If the option to *Check/exclude emails* has been selected, clicking in the *Next* button leads to a page where the list of


emails is shown and the user can select some of them to remove from the operation. If this option is not selected, the operation to send the emails is sent to a queue for processing. The browser will show the record that contains the information about the status of this request.

Making personalized content available to learners

Sending a personalized text is just one possible way to make this content available to learner. Another one is to offer the content through a URL that can be given to the learners. To enable such URL click in the icon with three dots in the right most corner of a personalized text action.

	Midterm comments 	Comment the midterm scores divided by topic	—	 Run 
---	--	---	---	---

You will see an operation labeled URL followed by either the word (Off) or (On). Select that operation. The following window shows the URL in which the content is available as well as the field to enable/disable it.

URL to the personalised content 

This URL provides access to the content personalised for each user.

`http://localhost:64709/action/26/serve/`

☐ URL available to users?

Action available from

Action available until

Access is protected by authentication, and the user **email** is in a column named **email**. To use a different column, add this parameter to the end of the URL:

`?uatn=[COLUMN NAME]`

replacing `[COLUMN NAME]` by the colum to use instead of **email**.

✓ Ok

In order for the learners to be able to view their personalized content, they have to be users of the OnTask platform and their ID present in the data table. This functionality is conceived for a context in which OnTask authenticates users either through a corporate Single-sign on layer, or learners access the OnTask through the Learning Management System with a LTI interface (see [Authentication](#)).

Running a survey

After creating a *survey action* it can be used in two modalities: run by the instructor, or given to the learners to fill out the data. The first modality is used as a mechanism to capture instructor observations. For example, surveys run by the instructor can be used as an attendance capturing mechanism (if the instructor has a device or procedure to capture who is in attendance). If the *Run* operation is selected, OnTask shows a table with the learners selected for the action, and the values for the survey collected so far.

BIOL1011

Student comments Week 1 [Edit](#)

Feedback about today's lecture

Show 10 entries Search:

email	Difficult Week 1	Sat W1
ckrn7263@bogus.com		
ctfh9946@bogus.com		
euho9752@bogus.com		
euhz6532@bogus.com		
isim6886@bogus.com		
kjb6748@bogus.com		
olie8242@bogus.com		
pzaz8370@bogus.com		
tcnf7608@bogus.com		
tdrv2640@bogus.com		

Showing 1 to 10 of 14 entries

Previous **1** 2 Next

[Back](#)

© OnTask Project 2019 Version: B.4.0.1

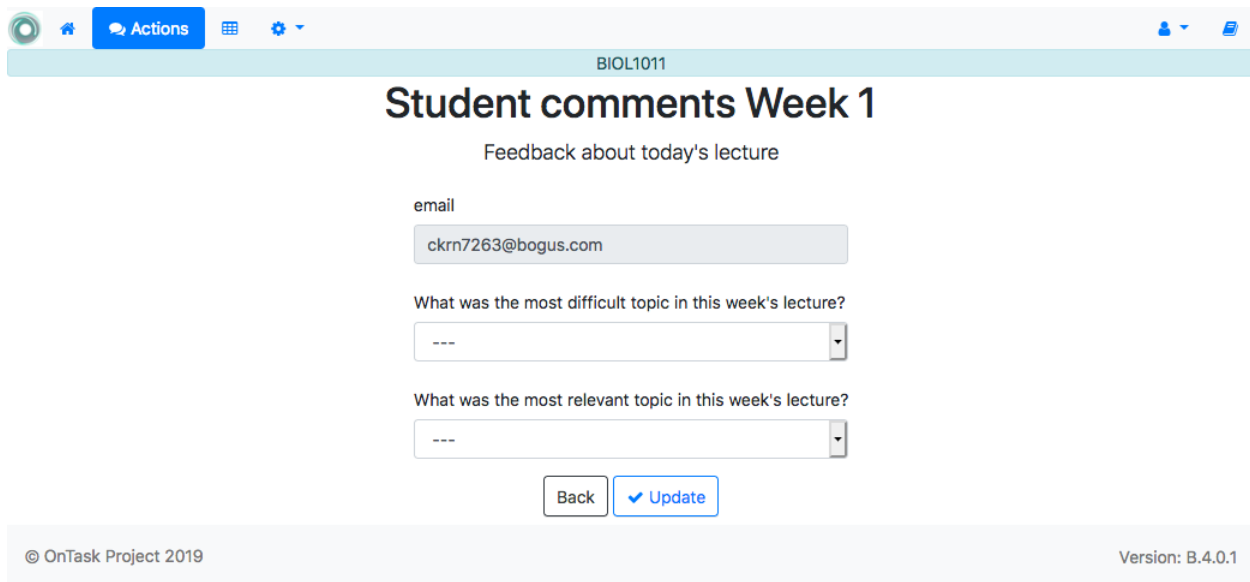
Each row contains the identifier of the student (in the previous table, the email) as a link. Instructors may click in a link available to enter the survey information or modify the already existing information for that learner.

After entering the information the list of students for which the data entry is still allowed.

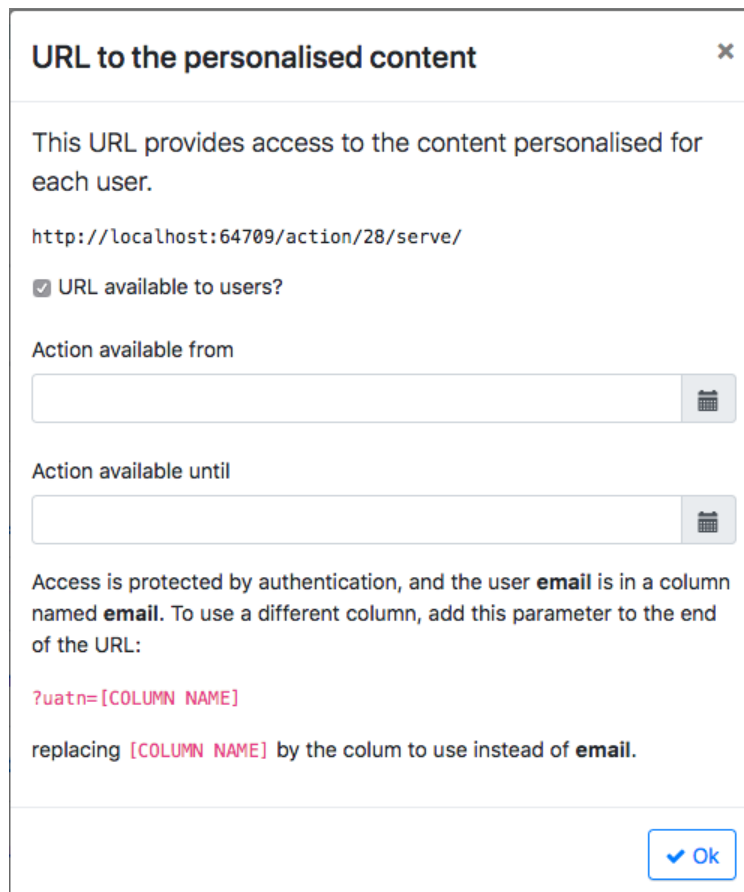
Making the survey available to the learners

The second operation available for *survey actions* is to make available the URL to learners so that they individually enter the information themselves. The right-most column of the action table contains an icon with three dots that if selected shows a set of additional operations, and one of them has the text *URL*. If selected OnTask shows the URL for the survey, the possibility of enable/disable it, or even provide a date/time window for its availability.

Once enabled, you may send the URL to the students (you may even use a personalized text action for that). Once the students click in the action, and if they are allowed to connect to OnTask as basic users, after authentication, if



The screenshot shows the OnTask web interface. At the top, there is a navigation bar with a home icon, an 'Actions' button, a calendar icon, a settings icon, and a user profile icon. Below the navigation bar, the course 'BIOL1011' is displayed. The main heading is 'Student comments Week 1', followed by the subtitle 'Feedback about today's lecture'. The form includes an 'email' field with the value 'ckrn7263@bogus.com'. There are two dropdown menus: 'What was the most difficult topic in this week's lecture?' and 'What was the most relevant topic in this week's lecture?'. At the bottom of the form are 'Back' and 'Update' buttons. The footer contains '© OnTask Project 2019' and 'Version: B.4.0.1'.



The screenshot shows a modal dialog titled 'URL to the personalised content'. It contains the following text: 'This URL provides access to the content personalised for each user.' followed by the URL 'http://localhost:64709/action/28/serve/'. Below the URL is a checkbox labeled 'URL available to users?' which is checked. There are two date pickers: 'Action available from' and 'Action available until'. Below these is a paragraph: 'Access is protected by authentication, and the user **email** is in a column named **email**. To use a different column, add this parameter to the end of the URL:'. This is followed by the text '?uatn=[COLUMN NAME]' and 'replacing [COLUMN NAME] by the column to use instead of **email**.' At the bottom right is an 'Ok' button.

their email is part of the table, they will see a form with the survey questions and after answering it, the values are automatically stored in the right row and column in the table.

These two survey actions are ideal to collect information about any aspect of a learning experience in a way that is centralized and available for further processing through personalized text actions. For example, users may choose from a predefined set of topics those that were more challenging. This information can then be used in a personalized text action to provide the adequate resources to each learner.

Sending personalized emails in Canvas

The execution of a *Personalized Canvas Email* action requires additional information as shown in the following figure:

BIOL1011

Send Canvas Email for action "Send Canvas reminder"

2 Emails will be sent.

Column with the Canvas ID*

Email subject*

Canvas Host*

Name of the Canvas host to send the messages

☐ Check/Exclude Canvas IDs before sending?

☐ Download a snapshot of the workflow?

A zip file useful to review the emails sent.

Cancel Preview > Next

© OnTask Project 2019 Version: B.4.0.1

Column in the table containing Canvas ID values This column is essential to send the emails to the platform as it is used to uniquely identify every Canvas user. The data can be obtained downloading the marks of a course as a CSV file and uploading/merging this column with the existing table.

Email subject The text to use as subject for the messages.

Canvas Host If there is more than one Canvas platform configured in OnTask, an additional pull-down menu will appear to select which one to use. If there is a single Canvas platform configured, it will be used by default and this field will not be shown in the form.

Check/Exclude Canvas IDs before sending messages If this option is selected OnTask adds an extra step to check the identifiers that will be used and offer the possibility of excluding some of them.

Download a snapshot of the workflow If selected, a snapshot of the workflow (data and actions) will be downloaded after the messages have been queued for delivery.

After introducing this data, OnTask will check if it has credentials for the user to access Canvas through its API. If not, the user will be redirected to a page in the Canvas Learning Management System to 1) authenticate, and 2) authorize OnTask to access the platform. If these steps are successful, the user is redirected back to OnTask and the messages are delivered. The credentials retrieved from Canvas will be reused for future executions of this action.

Sending personalized JSON objects to another platform

The operation to *Run* a personalized JSON action sends the objects resulting from the personalization to the given URL. The page to collect the information to run these actions is shown in the next figure:

The screenshot shows the OnTask web interface. At the top, there's a navigation bar with 'Actions' highlighted. Below it, a header bar displays 'BIOL1011'. The main heading is 'Send JSON objects for action "Send JSON to remote server"'. A sub-message states '14 JSON objects will be sent.' Below this, there's a form with two main fields: 'Column to exclude objects to send (empty to skip step)' with a dropdown menu showing '---', and 'Authentication Token*' with a text input field. A note below the token field says 'Authentication token provided by the external platform.' At the bottom of the form are three buttons: 'Cancel', 'Preview' (with an eye icon), and '> Next'. The footer of the interface shows '© OnTask Project 2019' on the left and 'Version: B.4.0.1' on the right.

The first field is the column to perform a last review of the elements to send and select some of them to exclude in an extra step. If the field is empty, this step is skipped. The second field is the token to use for authentication when sending the JSON objects to the URL given when editing the action. This operation assumes that such token has already been obtained and provides no additional functionality to execute that step as part of this operation.

Similarly to the email actions, once these fields are provided, the operation to send the JSON objects to the target URL is queued in a batch system for processing. The browser shows the record where the status of this request is reflected.

Creating a ZIP file with the personalized text

The *personalized text actions* offer the possibility of creating a ZIP file containing one HTML file with the personalized text for every learner. The execution of this operation requires the use of two columns in the table and a suffix to create the file names. The operation is available clicking in the icon with three dots in the right-most column of an action in the action page. The additional information is requested through the form shown in the following figure.

The first part of the file name is taken from the values of a key column. The second part of the file name is taken from a second column (optional). Additionally, the user may include a third suffix to be used for the last part of the file name (if none is given the default suffix is `feedback.html`). For example if the first column has the values `submission 01`, `submission 02`, `submission 03`, the second column has the names `John`, `Paul`, `Mary`, and the file suffix is empty, the ZIP file will be created with three HTML files with names `submission 01_John_feedback.html`, `submission 02_Paul_feedback.html` and `submission 03_Mary_feedback.html`.

Uploading feedback files for a Moodle Assignment

One of the potential uses of the ZIP file generated from a personalized text action is to upload each file as personalized feedback of an assignment in a Moodle course. However, there are some requirements in the file names so that they are uploaded each to the appropriate location, namely:

1. The table must have column named `Identifier` with values starting with the word `Participant` followed by a white space and a unique number. This column can be extracted from a Moodle Assignment by downloading the *grading worksheet*:



BIOL1011

ZIP for action "Midterm comments"

A ZIP with 14 files will be created.

Key column to use for file name prefix (Participant id if Moodle ZIP)*

Column to use for file name prefix (Full name if Moodle ZIP)

File name suffix ("feedback.html" if empty)

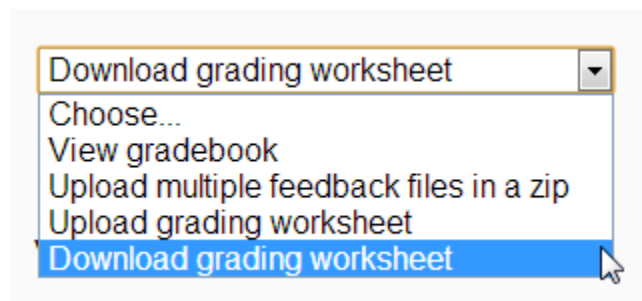
☐ This ZIP will be uploaded to Moodle as feedback

☐ Check/exclude users before sending?

[Cancel](#) [Preview](#) [Next](#)

* This function creates a ZIP with one HTML file for each user selected by the action. The file contains the personalised text as shown when using the "Preview" button. The file name is created with the values of the two selected columns followed by the suffix. If you plan to use the ZIP to use as file feedback in Moodle, the first column must contain values of the form "Participant [number]" and the second must be the user's full name. These columns can be obtained by downloading first the grading CSV file from the Moodle assignment, merge them into the workflow, and then use them to create this ZIP and upload it.

© OnTask Project 2019 Version: B.4.0.1



The CSV file has two columns with names Identifier and Full name.

2. The two columns Identifier and Full name must be *merged* with the current data in the workflow.
3. Choose the column Identifier and Full name as the first and second column respectively when generating the ZIP file. Make sure you select the option This ZIP will be uploaded to Moodle as feedback.
4. Upload the resulting ZIP using the option Upload multiple files in a zip in the Moodle Assignment.

1.3.5 Workflow Settings

In addition to the operations described in the previous sections to upload and merge data, create actions, and run these actions, there are additional operations for the workflow that can be accessed through the Settings pull-down menu in the top-bar menu. The settings pages offer information about: workflow operations (export, rename, clone, etc.), column operations (change column type, rename, clone, etc.), scheduled actions, and view logs.

Workflow Operations

Export Rename Clone Flush data table Delete

Attributes Share

Show 10 entries Search:

Name	Value	
Coordinator name	Sarah Jones	
Course name	Cell Biology	

Showing 1 to 2 of 2 entries Previous 1 Next

© OnTask Project 2019 Version: B.4.0.1

Export Workflow

Number of rows/columns: 14/67

Number of actions: 6

Select the actions to export:

Action	All
Badges	<input type="checkbox"/>
Comments about how to prepare the lecture (Week 4)	<input type="checkbox"/>
Midterm comments	<input type="checkbox"/>
Send JSON to remote server	<input type="checkbox"/>
Student comments Week 1	<input type="checkbox"/>
Suggestions about the forum	<input type="checkbox"/>

Cancel Export

© OnTask Project 2019 Version: B.4.0.1

Rename workflow ✕

Name*

Description text

Confirm data deletion ✕

Are you sure you want to flush the data in workflow BIOL1011?

This operation will also remove actions, table views and records.

Confirm workflow deletion ✕

Are you sure you want to delete the workflow BIOL1011?

This operation will also remove all the conditions and actions attached to it.

In addition to the buttons at the top, the *Workflow Operations* page offers two additional operations shown in separated tabs.

Attributes This tab shows the operations to manage a dictionary of pairs (name, value). The names can be used in personalized text and they are replaced by the value.

The screenshot shows the 'Workflow Operations' page for 'BIOL1011'. The 'Attributes' tab is active, displaying a table of attributes. At the top, there are buttons for 'Export', 'Rename', 'Clone', 'Flush data table', and 'Delete'. Below the buttons, there is a '+ Attribute' button and a search bar. The table has two columns: 'Name' and 'Value'. The 'Name' column contains 'Coordinator name' and 'Course name', both with edit icons. The 'Value' column contains 'Sarah Jones' and 'Cell Biology', both with delete icons. The page also shows 'Showing 1 to 2 of 2 entries' and pagination controls (Previous, 1, Next). The footer indicates '© OnTask Project 2019' and 'Version: B.4.0.1'.

Name	Value
Coordinator name	Sarah Jones
Course name	Cell Biology

The *Attribute* button opens a form to introduce the name and value of a new attribute. The table below this button shows the attributes available for the workflow. The name of the attribute is a link that opens the form to edit its content. The icon on the right deletes the attribute.

The use of these attributes is to facilitate a single point of change when a value appears in multiple locations. For example, every personalized text action is signed with the name of the instructor. If that name changes, all actions need to be edited. On the other hand, if the actions contain the name of an attribute, just changing the attribute value propagates the change to all actions.

Share The *Share* tab on the right of the page is used to manage the list of users that have access to this workflow. The *User* button opens a form to introduce the id of the user to share the workflow.

Column Operations

The *Column Operations* page is available through the Settings link at the top-bar menu. Columns in OnTask have substantial information and operations that is condensed in this page. The information is shown as illustrated by the following figure.

The buttons at the top of the page offer the following operations

Column This menu opens three options: create a regular column, create a column combining the values of two existing columns, or create a new column with random values.

Add a regular column Opens a dialog to create a new column in the table with the following fields:

- Name (mandatory): column name (shown in the table)
- Description: text that will be shown to the learners if the column is part of a survey action.

BIOL1011

Workflow Operations

[Export](#)
[Rename](#)
[Clone](#)
[Flush data table](#)
[Delete](#)

Attributes Share

Users with access to this workflow

[+ User](#)

User	
instructor02@bogus.com	
instructor03@bogus.com	

© OnTask Project 2019 Version: B.4.0.1

BIOL1011

Column Operations

[+ Column](#)
[Manage table data](#)

Show entries Search:

#	Name	Description	Type	Key?	
1	SID		%	✓	...
2	email		I	✓	...
3	Surname		I		...
4	GivenName		I		...
5	MiddleInitial		I		...
6	Gender		I		...
7	UOS Code		I		...
8	Program		I		...
9	Enrolment Type		I		...
10	Attendance		I		...

Showing 1 to 10 of 67 entries Previous **1** 2 3 4 5 6 7 Next

You may drag and drop the columns from the left-most cell to change their position in the workflow.

© OnTask Project 2019 Version: B.4.0.1

Add column ✕

Column name*

Description

Type of data to store in the column*

number ▼

Column position (zero to insert last)*

0 ▲ ▼

Column active from

📅

Column active until

📅

Comma separated list of values allowed in this column

Value to assign to all cells in the column

Cancel

✓ Add column

- Data type (mandatory): The possible data types are *number* (representing both integers or real numbers), *string*, *boolean* (only possible values are *true* and *false*), and *datetime* (a date and time together).
- An integer (mandatory) representing the position of the column in the table (a value zero will insert it at the end of the table).
- Two date/time values to control the visibility of the column.
- Comma-separated list of possible values. This field is to restrict the values in the column. The values have to be compatible with the specified data type.
- Initial value to assign to all cells in the column.

Add a formula-derived column This column is created by combining the values of existing columns using one of the operations addition, product, maximum, minimum, mean, median, standard deviation, conjunction or disjunction. The formula is only applied when the column is created the current values of the other columns. The column is not refreshed if the operand change in the future.

Add a column with random values This is useful to create columns for A/B testing. The new column is created with a random value from either a numeric range (starting at 0) or a set of strings.

Manage table data Menu to upload/merge data to the table (see [Data](#) for a detailed explanation), execute a plugin, or delete the data in the table.

Below the buttons to perform these operations the workflow columns are shown. If the number of columns is large, the information is divided into pages. The field at the top right of this list performs searches in all fields of all columns (name, type, etc). Each column has the following information (from left to right):

Position A number starting at 1. The position is used when [visualizing the table](#).

Name Unique name for the workflow that cannot contain the quotes ' or ".

Type One of integer, string, double, boolean or date/time.

Key? Field stating if it is a **key column**, that is, the values are different or unique for all rows. OnTask detects automatically this property when the data for a new column is loaded. You may edit and change this properly as long as the two requirements are satisfied: the values have to be unique for all rows, and there must be at least one key column per workflow.

Operations The operations available for columns are:

Edit Available by clicking in the column name. It allows you to change the name, unique and values allowed in the column. If the field *Has unique values per row* property is modified, OnTask checks if the values satisfy this condition.

The column may also have a *validity window* defined by two date/times. This validity is used when executing *action in tasks*.

Clone Clones the column in the workflow changing its name adding the prefix *Copy of* to the name.

Statistics (only for non-key columns) Shows a statistical summary of the values in the column. If the data type is *number*, the summary includes information about quartiles, a boxplot, and a histogram. For the rest of data types, the summary only includes the histogram.

Restrict Assigns as *allowed values* for the column those currently stored. This operation is useful to transform a generic column into one with values limited to the current ones.

Delete Deletes the column from the workflow. If there are conditions in the actions that use this column, those conditions will be removed from the action.

Make first column Move this column to the first position in the workflow

Make last column Move this column to the last position in the workflow

Edit column SID ✕

Column name*

Description

Type of data to store in the column*


number

Column position (zero to insert last)*


1

☒ Has unique values per row

Column active from



Column active until



Comma separated list of values allowed in this column

Cancel

✓ Update

The position of the columns can also be changed by dragging by the number (left-most column) and dropping them in the new position.

Scheduled Actions

“I have no regular schedule. I get up whenever I can.” – Jimmy Wales

The *personalized text*, *personalized canvas email* and *personalized JSON object* actions can be scheduled to run at some point in the future. To schedule the execution of an action go to the *Actions* page from the top menu, click in icon with three dots in the right-most column of the action and select the operation *Schedule*.

Scheduling a Personalized Text Action

The following figure shows the information requested to schedule the execution of a personalized text action (sending emails to learners):

BIOL1011

Schedule Action Execution

Time now is Feb. 19, 2019, 11:57 a.m.

Midterm comments

Name*

Send Emails after week 3

Description

When to execute this action*

2110-07-05 17:30:51

Column in the table containing the email*

email

Email subject*

Your preparation activities for the week

Comma-separated list of CC Emails

Comma-separated list of BCC Emails

☐ Check/exclude email addresses before scheduling?

☐ Send you a confirmation email

☒ Track if emails are read?

Cancel Next

© OnTask Project 2019 Version: B.4.0.1

The fields in this form are:

Name A name to identify this scheduling (a user may have several of these actions pending in a workflow)

Description A brief description explaining this scheduled action (for example, “send reminder before the exam”)

Column containing email The column in the table used to fill out the destination email. OnTask will check that the values in that column are proper email addresses.

When to execute the action A date/time in the future when the action will be executed.

Email subject The text to be included in the email subjects.

Comma separated list of CC emails A comma separated list of emails to include in the *carbon copy* (or CC) field of the email.

Comma separated list of BCC emails A comma separated list of emails to include in the *blind carbon copy* (or BCC) field of the email.

Send confirmation email Select this option if you want a confirmation email sent to you.

Track when emails are read Try to detect if the email is read. OnTask adds an extra column to the table to store the number of times the message is opened. This detection relies on how the email client processes the message, therefore, the information in this column may not be accurate.

Check/exclude emails If selected, this option inserts an extra step showing the emails and offering the possibility to eliminate them from the action. This option is useful to perform a final check and remove emails that cannot be removed with the action filter.

Scheduling a Personalized JSON Action

The following figure shows the information requested to schedule the execution of a personalized JSON action (sending JSON object to another platform):

The fields in this form are:

Name A name to identify this scheduling (a user may have several of these actions pending in a workflow)

Description A brief description explaining this scheduled action (for example, “send reminder before the exam”)

Column to select elements A column to show its values and allow to review and exclude some of the entries. This option is useful to perform a final check and remove entries that cannot be removed with the action filter.

Authentication Token The string to be use to authenticate with the external platform.

Table with Scheduled Actions

The table showing all the action scheduling operations can be access through the *Settings* link in the top-bar menu selecting the *Scheduled actions* link. The list of scheduled actions is shown as illustrated in the following figure.

The operations available for each operation are:

Edit This operation is accessed clicking in the name of the scheduled operation.





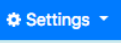
Action name The link in this field opens the edit page for the action.

Scheduled Time/date when the executing will take place

Status Status of this execution

View details View all the details for this scheduled execution (includes values in the payload)

Delete Delete the scheduled execution.

BIOL1011

Schedule Action Execution

Time now is Feb. 19, 2019, 11:57 a.m.

Send JSON to remote server

Name*

Description

Column to select elements (empty to skip)






☐ Check/exclude email addresses before scheduling?

When to execute this action*

Authentication Token*

Authentication token provided by the external platform.

© OnTask Project 2019
 Version: B.4.0.1

BIOL1011

Scheduled Actions*

Show entries

Search:

Name	Action	Scheduled	Status	
Send Emails after week 3	Midterm comments	07/05/2110 5:30 p.m.	Pending	
Send JSON object in Week 5	Send JSON to remote server	07/25/2110 5 p.m.	Pending	

Showing 1 to 2 of 2 entries

Previous **1** Next

*The execution of some of the actions can be scheduled for a time in the future. This page shows those planned executions. To schedule the execution of one action, go to the [Actions](#) menu and select the operation *Schedule* for an action.

© OnTask Project 2019
 Version: B.4.0.1

View logs

The platform keeps a log of most of the operations that are executed when managing a workflow. These records are available through the *View Logs* link in the *Settings* pull down menu at the top-bar menu. The page shows information about the records in tabular form. The following figure shows an example.

BIOL1011

Logs*

CSV Download

Show 10 entries Search:

ID	Date/Time	User	Event type
63	2018-09-15 19:21:55 +0930	instructor01@bogus.com	workflow_share_add
62	2018-09-15 19:21:44 +0930	instructor01@bogus.com	workflow_share_add
61	2018-09-15 18:48:54 +0930	instructor01@bogus.com	workflow_import

Showing 1 to 3 of 3 entries Previous 1 Next

*The platform records all operations performed on the workflow as *Logs*. These elements are shown in the table above. You can view its content by clicking on its number.

© OnTask Project 2019 Version: B.4.0.1

The *CSV Download* button allows to download the logs in CSV format for further processing. Additionally, the content of the table is paginated and the links to access each page are shown at the bottom. The records can be searched using the box at the top right corner of the table.

1.3.6 Plugins: Transforming the data with your own code





The additional method offered by OnTask to manipulate the data in a workflow table is to execute arbitrary Python code encapsulated as a Python module and placed in a predefined folder in the computer hosting the server. In the context of the platform, these Python modules are called **Plugins** or **Transformations** and require some *previous configuration*. Before their execution, a plugin must be written and installed in the folder previously considered for that purpose.

The purpose of the plugins is to allow arbitrary transformations of the data attached to a workflow such as machine learning algorithms, predictive models, etc. The list of plugins available for execution can be accessed through the link *Execute plugin* inside the drop-down menu *Manage table data* available in both the *Actions* and *Table* pages. The transformations available for execution are shown in a table like the one in the next figure.

Each transformation is shown with a the name of the internal folder in which it is installed, its (unique) name, a description, if it is ready to run, and the last time the code was modified (based on the file modification time). The link in the name opens a form to introduce the information required for execution. The following figure shows an example of this page.

The information requested in this page is divided into three tabs.












Columns to pass to the transformation This field is to select the subset of columns from the data table that will be passed when invoking the transformation. It is possible for a transformation to define a set of *fixed* column names as inputs. If this is the case, the field in this tab shows those names and does not allow changes.




Table


BIOL1011





Transformations

Show entries Search:

Folder 	Name 	Description 	Ready to run 	Last executed 
linear_model	Linear Model 	Plugin to execute a linear model. The required parameters are: 1) List of input columns to use in the linear model (c1, c2, ..., ck) 2) Comma-separated list of coefficients for those columns (q1, q2, ..., qk) 3) Constant: p 4) Name of the result column "result" The plugin then computes: $result = p + c1 * q1 + c2 * q2 + \dots + ck * qk$	✓	--
mslq	MSLQ Score calculation 	Plugin to calculate the scores of MSLQ The names of the columns must be MSLQ_Q01 to MSLQ_Q44.	✓	--
spq	SPQ Score calculation 	Plugin to calculate the scores of SPQ The names of the columns must be SPQ_Q01 to SPQ_Q44.	✓	--
srolq_r	SROLQ-R Score calculation 	Plugin to calculate the scores of SROLQ-R The names of the columns must be SROLQ-R_Q01 to SROLQ-R_Q44.	✓	--
test_plugin_1	Test Plugin 1 Name 	Test Plugin 1 Description Text	✓	--
test_plugin_2	Test Plugin 2 Name 	Test Plugin 2 Description Text	✓	--

Showing 1 to 6 of 6 entries Previous **1** Next

© OnTask Project 2019 Version: B.4.0.1




Table


BIOL1011


Transformation "Test Plugin 1 Name"

Columns to pass to the transformation

Columns to store the result


Parameters

Input Columns (to read data)



To select a subset of the table to pass to the plugin

Cancel

Run 

© OnTask Project 2019 Version: B.4.0.1

Columns to store the result The middle tab in this page includes fields to obtain the output column names (the transformation may supply suggestions, an optional suffix to add to the result column names to be able to differentiate between multiple executions of the transformation, and a key column to be use when merging the result of the plugin with the current table.

Parameters This tab contains a form to pairs (*name*, *value*) as defined by the transformation.

After the data has been provided the tool shows an transformation execution report showing the columns that will be created and how will they be merged with the existing data.

Plugin requirements

The Python modules installed in the predefined folder need to satisfy various requirements to be considered for execution within OnTask. More precisely, each plugin must be stored in its own folder (as a Python module). The file `__init__.py` in the module must contain:

1. Module variable `class_name` with the name of the class in the file that contains the required definitions.
2. The definition of a class with the name stored in the previous variable.
3. Class field `name` with the plugin name to show to the users.
4. Class field `escription_txt` with a string with the detailed description of what the plugin does
5. Class field `input_column_names` with a potentially empty list of column names (strings). If the list is empty, the columns are selected by the user at execution time.
6. Class field `output_column_names` with a non empty list of names (strings) of the columns to be used for the output of the transformation.
7. Class field `parameters` with an optionally empty list with tuples with the following structure:

```
('name', type, [list of allowed values], initial value, help_text)
```

These elements will be requested from the user before executing the plugin through a form. The conditions on these values are:

- name must be a string
 - type must be a string equal to “integer”, “double”, “string”, “datetime” or “boolean”.
 - The list of values is to restrict the possible values
 - The initial value must be of the type specified by the second element.
 - Help_text a string to show as help text
8. Class method `run` that receives:
 - a pandas data frame with the data to process
 - a string with the name of the key column that will be used to merge the result.
 - A dictionary of pairs (name, value) with the parameters described in the previous element.

and returns a result Pandas data frame. This frame **must** have one column with the key column name provided so that it can be properly merged with the existing data.

If a plugin does not comply with these properties the platform shows a summary of these checks to diagnose the problem.

Diagnostics ×	
<button>Close</button>	
Status	Test
Ok	Presence of a string field with name "name"
Ok	Presence of a string field with name "description_txt"
Ok	Presence of a field with name "input_column_names" storing a (possible empty) list of strings
Not found	Presence of a field with name "output_column_names" storing a non-empty list of strings
Unchecked	Presence of a (possible empty) list of tuples with name "parameters". The tuples must have six elements: name (a string), type (one of "double", "integer", "string", "boolean", or "datetime"), (possible empty) list of allowed values of the corresponding type, an initial value of the right type or None, and a help string to be shown when requesting this parameter.

1.4 Use Scenarios

This section contains a set of examples or *scenarios* showing how OnTask can be used. If you prefer to read about use cases instead of the manual, this is your place.

The first question you have to answer is if it really makes sense to use this tool in your context or is this yet another useless software application. To identify if OnTask is useful answer the following questions:

1. Is there a moment during the deployment of a learning experience in which it makes sense to either gather or provide students with personalised information about their experience?
2. Do I have access to the data to personalise such communication with the students? For example, if you want to contact certain students regarding the use of certain resource, do you have the data telling you who used such resource?
3. Do I know what I would like to do once I got both the moment and the data identified? For example, I would like to write certain students a message, or provide them some specific resources, or give them some suggestions.

OnTask will likely be useful to you if you answer **yes** to all three questions. It is very important that you start your process from the learning design, that is, from where in the experience you think makes sense to provide some personalised resource to the students. In the remaining of this section we have included a set of scenarios that range from very simple to very complex. Each of them has two files available for you to download:

1. The CSV file with the data used in the example.
2. The complete workflow (data and actions).

Create a new workflow and use the first file to upload the data and try to replicate the steps described in each scenario, or use the second file to import the complete workflow and browse through its elements.

1.4.1 Scenario? I don't even know what the tool does

- CSV data file: `Scenario 1 Data File`.
- Workflow data file: `Scenario 1 Workflow File`.

Suppose you have a course with 500 students and they take the course as part of four programs named FASS, FEIT, FSCI, SMED in different disciplines. They are all related, but at the same time different. You would like to send them a text explaining the connection between the material studied in the course and their discipline. The task would then consist of drafting four email texts each of them explaining the connection respectively with FASS, FEIT, FSCI and SMED, and then send the appropriate message to each student.

Suppose now that you do have the class listing in an excel sheet. The table has the names of the students (one student per row), and the column named *Program* contains for each student one of four possible values “FASS”, “FSCI”, “FEIT” or “SMED”.

Steps

1. Create a new workflow (click in the *New Workflow* button in the home page of the tool). Introduce a name and a description (optional) for this workflow.
2. Open the newly created workflow by clicking its name. A page titled *Workflow Details* will be shown.
3. The page says that there is no data uploaded yet, and to upload it you need to click in the *Dataops* link either at the bottom of the text or at the context menu at the top.
4. A new page titled *Upload/Merge* appears. Select the option *CSV Upload/Merge* and follow the instructions to upload the `CSV file` (select all the columns to upload)
5. Click the *Details* page to see a summary of the elements in the workflow: 500 rows, 10 columns, 4 actions and 0 attributes. All the steps up to here can be done by importing the given workflow file.
6. Click the *Table* link at the top of the page and browse through the data included in the workflow. Locate the column called *Program* and verify that it has the correct values.
7. Click the *Actions* link at the top of the page. The new page contains a list of actions available for this workflow. Click on the button to create a *New Action Out* to send an email to the students in the SMED program. Choose an name and description (optional) for the action, for example *Email students in SMED*.
8. We now create a filter that selects only those students for which the column *Program* has the name *SMED*. Click in the *Add Filter* button at the top of the page and in the new screen introduce the elements shown in the following figure:

The filter has name *Students in MED* and the students are selected if the given *Formula* is true. In other words, if the *Program* column is equal to *SMED*. Create the filter.
9. Once the filter is created the top of the *Action Out* screen now shows how many students have been selected by the filter (in this example 129 out of 500).
10. Write now the greeting at the top of the text area and select from the pull-down menu labeled *Insert column name* the *GivenName*. You will see how the string `{{ GivenName }}` is inserted in the text. This is the way OnTask has to mark that the string with the double curly braces should be replaced by the value of the column *GivenName*.
11. Complete the rest of the message with a text explaining why the material and activities in this course are relevant to the *SMED* program. The *Action Out* screen should look similar to the one shown in the following figure:
12. Once you are satisfied with how the text read, click in the *Preview* button at the bottom of the page and you will see how the message is produced by changing the name of the student (the rest of the text remains the same). Use the arrows to traverse the list of 129 students in this group.
13. Save and close the *Action Out*. You go back to the screen showing you all the actions in the workflow (you can see the one we just created). If you click in the *Email* button, you will be asked for some extra data such as the email subject, the column to use as email address and some options to track email reading or even obtain a snapshot (duplicate) of the workflow as it is right now for future reference.

Edit filter ✕

Description

The learner will be selected if

NOT

AND

OR

+ Add rule

Add group

Program

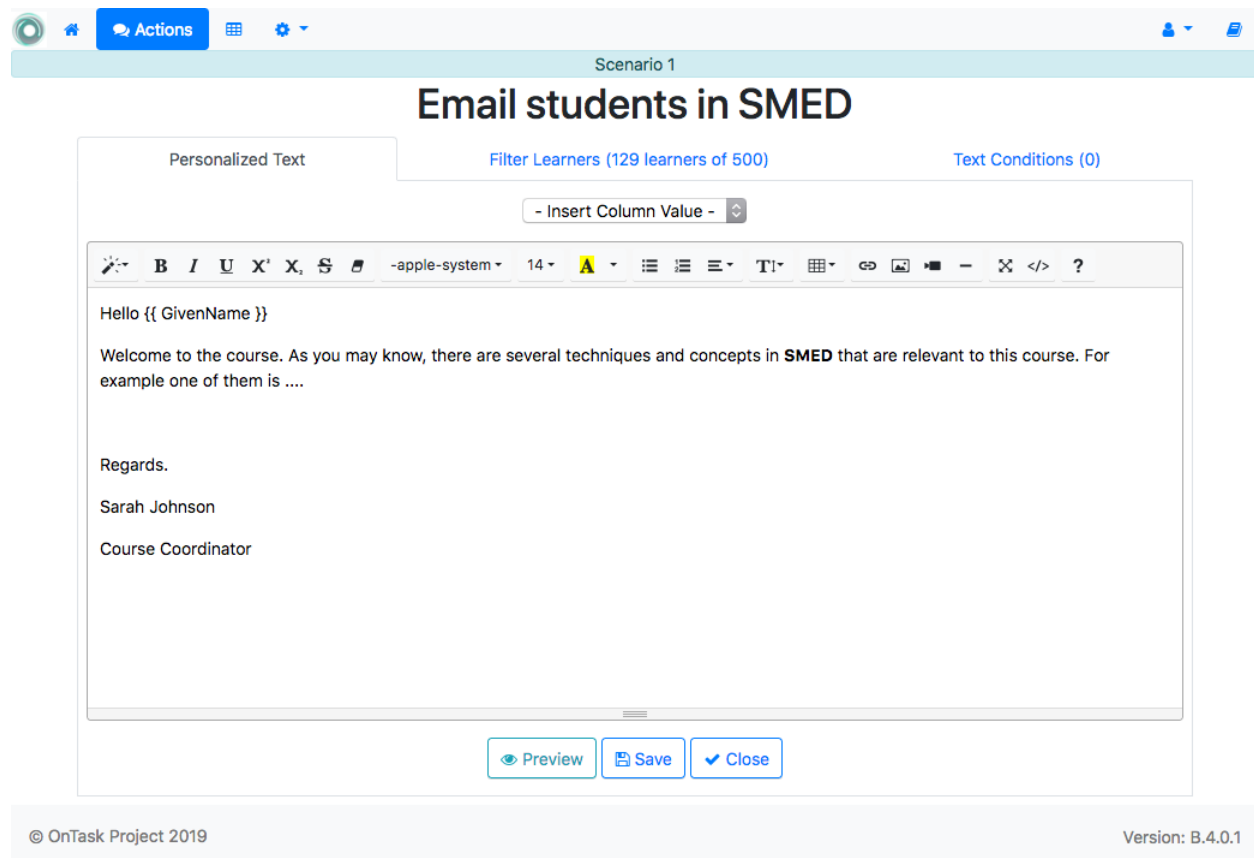
equal

✕ Delete

SMED

Cancel

Update



14. That's it! You now can create three additional similar messages to send emails to the students in the other three programs (FEIT, FASS and FSCI) with an almost identical rule (changing the condition in the filter and the email text).

1.4.2 Scenario 2: Send email to all students with tailored text

- CSV data file: Scenario 2 Data File.
- Workflow data file: Scenario 2 Workflow File.

Suppose you have a course with 500 students and they take the course as part of four programs named FASS, FEIT, FSCI, SMED in different disciplines. They are all related, but at the same time different. You would like to send all your students an initial email explaining a few aspects of the course. One paragraph is about the structure of the course and is common to all students. A second one explains the connection between the material and their program. This second paragraph has to change depending on the program in which each student is enrolled. The task consists on drafting one email with a personalised greeting (using the student's name) a common paragraph, and a second paragraph that changes depending on the program.

Suppose now that you do have the class listing in an excel sheet. The table has the names of the students (one student per row), and the column named *Program* contains for each student one of four possible values "FASS", "FSCI", "FEIT" or "SMED".

Steps

Scenario 1

Send emails for action "Email students in SMED"

129 Emails will be sent.

Email subject*

Column to use for target email address*

email

Comma separated list of CC emails

Comma separated list of BCC emails

☐ Check/exclude email addresses before sending?

☐ Send you a summary message?

☐ Track email reading in an extra column?

☐ Download a snapshot of the workflow?

A zip file useful to review the emails sent.

Cancel Preview > Next

© OnTask Project 2019 Version: B.4.0.1

1. Create a new workflow (click in the *New Workflow* button in the home page of the tool). Introduce a name and a description (optional) for this workflow.
2. Open the newly created workflow by clicking its name. A page titled *Workflow Details* will be shown.
3. The page says that there is no data uploaded yet, and to upload it you need to click in the *Dataops* link either at the bottom of the text or at the context menu at the top.
4. A new page titled *Upload/Merge* appears. Select the option *CSV Upload/Merge* and follow the instructions to upload the *CSV file* (select all the columns to upload)
5. Click the *Details* page to see a summary of the elements in the workflow: 500 rows, 10 columns, 1 action and 0 attributes. All the steps up to here can be done by importing the given workflow file.
6. Click the *Table* link at the top of the page and browse through the data included in the workflow. Locate the column called *Program* and verify that it has the correct values.
7. Click the *Actions* link at the top of the page. The new page shows a list of actions available for this workflow. Click on the button to create a *New Action Out* to send a welcome email to all students. Choose an name and description (optional) for the action, for example *Welcome email*.
8. The next page is the action out editor. In the second area from the top you will see a button to create a new condition (*Conditions + New*). Click the button and a new window opens to specify a condition. Enter the following data:

The formula states that the condition will be satisfied when the column *Program* has the value *FASS*. Click the button to save the condition.

1. Create three additional conditions that are analogous to this one but are true when the *Program* column is equal to *FEIT*, *FSCI* and *SMED* respectively.

Edit condition ×

Name

Description

The text will be shown if

NOT AND OR

+ Add rule Add group

Program

equal

×

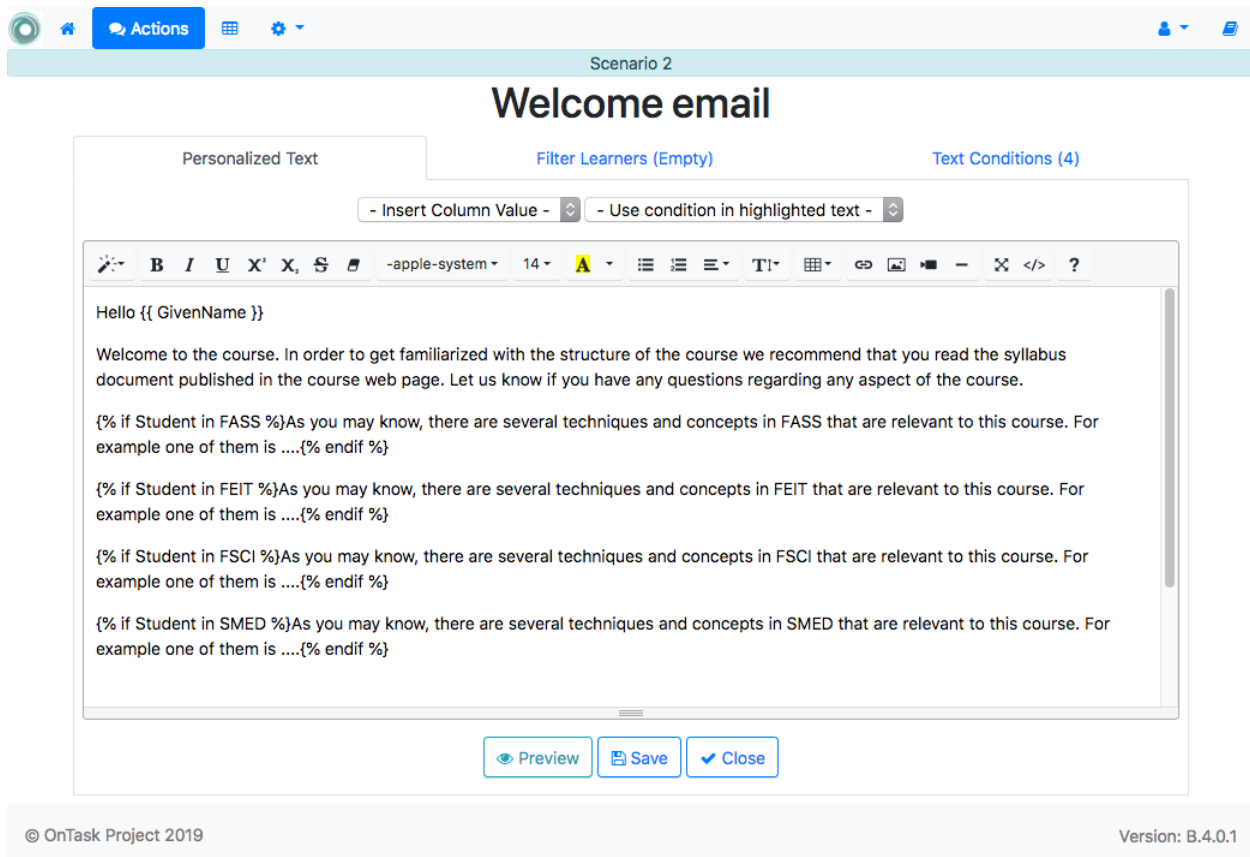
 Delete

FASS

Cancel

✓ Update condition

2. Write the text of the email starting with a greeting. Immediately after the greeting select from the pull down menu titled *Insert column value* the column *GivenName*. You will see how the string `{{ GivenName }}` is inserted in the text. This is the way OnTask has to mark that the string with the double curly braces should be replaced by the value of the column *GivenName*.
3. Next, write the four paragraphs that you would write to the students in each of the programs (FEIT, FSCI, FASS and SMED). Separate them with an empty line.
4. Now, and this is the crucial step, we are going to mark these paragraphs with one of the previously defined conditions. Select and highlight the paragraph to the students in the FASS program. Then click in the arrow next to the condition named *Student in FASS* and select the option *Insert in text*. The text should now be surrounded by the strings `{% if Student in FASS %}` and `{% endif %}`.
5. Repeat the same procedure now with the three remaining paragraphs. Select the text and then click in the appropriate condition. The final result should be similar to the one shown in the following figure.



6. Add the signature at the end of the text. Use the button named *Preview* at the bottom and use the arrows in the emerging window to browse through the different messages created for each user.
 7. Save and close the *Action Out*. You now are back to the screen showing the actions in the workflow (you can see the one we just created). If you click in the *Email* button, you will be asked for some extra data such as the email subject, the column to use as email address and some options to track email reading or even obtain a snapshot (duplicate) of the workflow as it is right now for future reference.
1. That's it! Each student will receive a message with the content of the second paragraph different depending on the program.

Preview 1 of 500

×

< Close >

Hello Alica

Welcome to the course. In order to get familiarized with the structure of the course we recommend that you read the syllabus document published in the course web page. Let us know if you have any questions regarding any aspect of the course.

As you may know, there are several techniques and concepts in FASS that are relevant to this course. For example one of them is

Regards.

Sarah Johnson

Course Coordinator

Values: Program = FASS

Scenario 2

Send emails for action "Welcome email"

500 Emails will be sent.

Email subject*

Column to use for target email address*

email

Comma separated list of CC emails

Comma separated list of BCC emails

☐ Check/exclude email addresses before sending?

☐ Send you a summary message?

☐ Track email reading in an extra column?

☐ Download a snapshot of the workflow?

A zip file useful to review the emails sent.

Cancel Preview > Next

© OnTask Project 2019 Version: B.4.0.1

1.4.3 Scenario 3: Send email to some students with personalised text

- CSV data file: Scenario 3 Data File.
- Workflow data file: Scenario 3 Workflow File.

Suppose you have a course with 500 students and they took a pre-test in the first week of the course. You have a CSV file with the class listing and a column with the pretest score. The column with name `Pretest` contains the score as a number between 0 and 100. You would like to send to those students that failed that pretest a personalized message with some suggestions about the material to revisit or improve. Additionally, these suggestions will be written differently depending on the score. You envision three messages. One encouraging more basic review for the scores below 30, one message a bit less strong for those with scores between 30 and 40, and a message based more on suggestions for those with scores between 40 and 50.

Steps

1. Create a new workflow (click in the *New Workflow* button in the home page of the tool). Introduce a name and a description (optional) for this workflow.
2. Open the newly created workflow by clicking its name. A page titled *Workflow Details* will be shown.
3. The page says that there is no data uploaded yet, and to upload it you need to click in the *Dataops* link either at the bottom of the text or at the context menu at the top.
4. A new page titled *Upload/Merge* appears. Select the option *CSV Upload/Merge* and follow the instructions to upload the CSV file (select all the columns to upload)
5. Click the *Details* page to see a summary of the elements in the workflow: 500 rows, 10 columns, 1 action and 0

attributes. All the steps up to here can be done by importing the given workflow file.

6. Click the *Table* link at the top of the page and browse through the data included in the workflow. Locate the column called *Pretest* and verify that it has the scores between 0 and 100.
7. Click the *Actions* link at the top of the page. The new page shows a list of actions available for this workflow. Click on the button to create a *New Action Out* to send the email with suggestions to the students that failed the pretest. Choose an name and description (optional) for the action, for example *Send email with suggestions*.
8. The next page is the action out editor. The are at the has the name *Filter* and is to define a condition to filter only a subset of the students. In this scenario, for this action, we want to consider only those students that have failed the pretest, or in other words, the pretest score is below 50. Click the *Add Filter* button and introduce this condition as shown in the following figure:

9. The next step is to define a condition to identify those students for which the score in the pretest is less than 30. In the second area from there is a button to create a new condition (*Conditions + New*). Click the button and a new window opens to specify a condition. Enter the condition as shown in the following picture:
10. Repeat the same operation (add another condition), but this time the condition is that the pretest score is larger or equal to 30 **and** lower than 40. The second condition can be added clicking in the *Add Rule* button. The result should be like the one shown in the following figure:
11. Create a third condition, but this time the rule should state that the pretest condition is larger or equal to 40.

Edit condition ✕

Name

Description

The text will be shown if

NOT AND OR

+ Add rule

Add group

Pretest

less

✕ Delete

30

Cancel

✓ Update condition

Edit condition ×

Name

Middle underperforming (30-40)

Description

The text will be shown if

NOT

AND

OR

+ Add rule

Add group

Pretest

greater or equal

30

×

Delete

Pretest

less

40

×

Delete

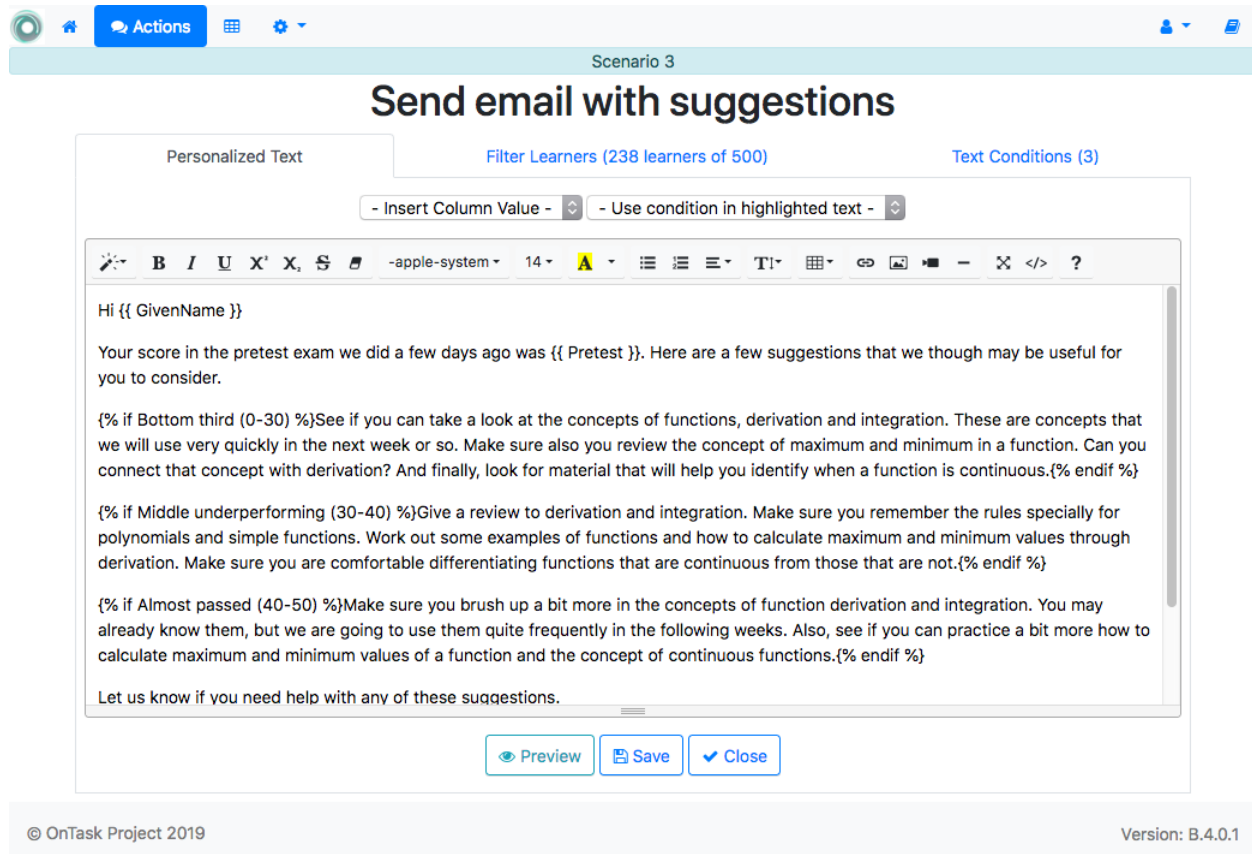
Cancel

✓ Update condition

80

Chapter 1. Version B.4.0.1

12. The next step is to write the body of the email starting with a greeting that includes the student's name and a sentence communicating the pretest score. Use the pull down menu labelled *Insert column name* to insert the column name.
13. Write three paragraphs for each of the categories of score. For each of them, select it, select the pull down menu of the appropriate condition, and click the option *Insert in text*. The text should be surrounded with some special mark-up such as `{% if Bottom third (0-30) %}` as shown in the following figure:



14. Use the preview button with the arrows to see how the different emails are shown (only for the students that satisfy the condition of the filter).
15. Save and close the *Action Out*. You now are back to the screen showing the actions in the workflow (you can see the one we just created). If you click in the *Email* button, you will be asked for some extra data such as the email subject, the column to use as email address and some options to track email reading or even obtain a snapshot (duplicate) of the workflow as it is right now for future reference.
16. That's it! Those students that failed the pretest will receive a message with some suggestions framed according to their score.

1.5 Tutorial

The tutorial is organized as a collection of tasks. Some of them require the use of a *set of files* that contain data, actions, and other additional elements. Before starting to work on the tasks make sure you have an instructor account in an OnTask server.

Actions

Scenario 3

Send emails for action "Send email with suggestions"

238 Emails will be sent.

Email subject*

Column to use for target email address*

email

Comma separated list of CC emails

Comma separated list of BCC emails

☐ Check/exclude email addresses before sending?

☐ Send you a summary message?

☐ Track email reading in an extra column?

☐ Download a snapshot of the workflow?

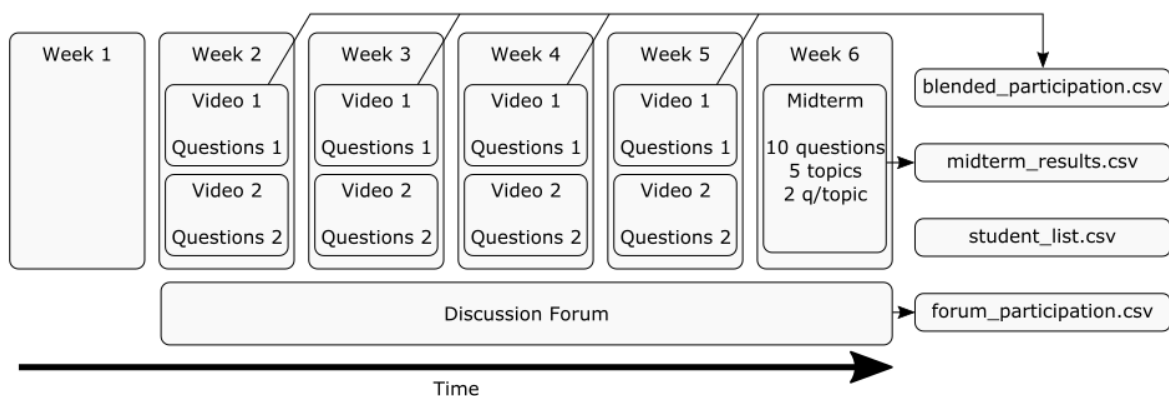
A zip file useful to review the emails sent.

Cancel Preview > Next

1.5.1 The Scenario

To make the activities in the tutorial as close as possible to the use of OnTask in a real situation all of them assume the following scenario.

A learning experience has started and is through its sixth week. The first week allowed learners to get acquainted with the course, objectives, structure, etc. No activities were proposed. From weeks 2 to 5 the same pattern of activities was used. Learners were asked to view videos and answer a complementary set of questions. In Week 6 a midterm examination took place consisting of 10 multiple-choice questions (about 5 topics, 2 question per topic). In addition to these activities, the course had during these weeks a discussion forum available for learners to ask questions, discuss matters related to the course or simply interact with other learners and instructors. The following figure illustrates the evolution of the course (the horizontal axis represents time from left to right)



During these six weeks *various data files* have been collected and made available to the instructors.

1.5.2 The Dataset

OnTask relies on the existence of data about how learners interact in a given experience. This activity offers an *artificially generated* data set to explore the functionality in the platform.

Download and unpack the zip file `dataset.zip` in a folder in your personal computer. The folder should have the following files: `student_list.csv`, `midterm_results.csv`, `forum_participation.csv`, `blended_participation.csv` and `all_data.csv`. These files have been derived from the *previously described scenario*.

This the information contained in each file:

File `student_list.csv` A file with information about 500 students and the following column names:

- Student id (SID),
- Identifier (an auxiliary field),
- email,
- Surname,
- GivenName,
- MiddleInitial,
- Full name,
- Gender,
- Course Code,
- Program, with one of the values *FSCI*, *FEIT*, *FASS* or *SMED*,
- Enrollment, with one of the values *HECS*, *Local* or *International*, and
- Attendance with values either *Full Time* or *Part Time*.

File `midterm_results.csv` File with information about 461 students with the following columns:

- SID (with values identical to those in the previous file),
- email,
- Last Name,
- First Name,
- Columns *Q01* to *Q10* with the result of the 10 multiple choice questions (1 means correct, 0 means incorrect), and
- the column *Total* with exam score (over 100 points).

File `forum_participation.csv` File with information about 500 student and their participation in the discussion forum in the course. The columns in this file are:

- SID (with values identical to those in the previous files),
- The columns *Days online*, *Views*, *Contributions* and *Questions* replicated four times for weeks 2-5 with the week number as suffix for the column name, and
- the accumulated values for *Days online*, *Views*, *Contributions*, and *Questions* without any suffix.

File `blended_participation.csv` File with information about learner engagement with the videos and questions complementing the videos for weeks 2 - 5 in the course. The columns in this file are:

- SID (with values identical to those in the previous files),
- Columns with names *Video_N_WM* contain the percentage of the video with number N in week M that has been visualized. For example *Video_2_W4* is the percentage of the second video in Week 4 that has been visualized.
- Columns with names *Questions_N_WM* contain the percentage of questions from group N in Week M that have been answered. For example, *Questions_1_W4* is the percentage of questions in block 1 from Week 4 that have been answered.
- Columns with names *Correct_N_WM* contain the percentage of questions from group N in Week M that have been correctly answered (and therefore a value smaller than the previous one).

File `all_data.csv` This file is simply the union of all the columns from the previous files.

Key columns

Each file in the dataset contains the data in *Comma Separated Values* or CSV. This format assumes that 1) the first line has the names of the columns separated by commas, and every line below that one contains the data for a row with the values also separated by commas. This format is used to store information with a structure similar to a table (rows and columns).

There is a special type of column that is of special interest in OnTask. If a column has a different value for each of the rows it is called a **key column**. The reason why these columns are important is because once one value is selected it unequivocally identifies one row in the table. For example, educational institutions typically assign an identifier (SID) to each student which is unique. If a table contains information about a set of students (one row, one student) and one column has the student ID, that column is then a **key column**. If during a procedure that is manipulating this table a student ID is given, that information uniquely identifies one row of that table.

1.5.3 Exploring the data with a spreadsheet

Open the file `student_list.csv` with a recent version of Excel. Select all the data in the spreadsheet (you can use `Ctrl-A` or `Cmd-A` in OSX) as shown in the following figure:

Next click in the menu item *Insert* and then click in the *Table* icon as shown in the following figure.

If you see a dialog showing you the range just selected and stating that the table has headers, just confirm the creation by clicking *OK*. You should now see the data in the spreadsheet with some coloring and a few icons in the cells at the top row as shown in the following figure.

The cells in the top row are the names of the columns contained in the file. If you click in the triangle at the right of any cell you will see a menu as shown in the following figure:

The menu allows you to sort the rows according to the value in the column (the buttons *Ascending* and *Descending*) and *filter* or select some of the rows to be viewed. In the example, the column contains four values that you can select individually. Click in the bottom part of the window to select/discard values and verify that the content of the table changes (only a subset of rows is shown). You may view all rows by choosing the item (*Select All*). Repeat this procedure with the columns with names *Program*, *Enrolment Type*, *Attendance*, *Gender* and *Course Code*. Once you select a value with the filter, type `Ctrl-up` (`CMD-up` in OSX) to move to the top of the table. Hold the Shift key and press `Ctrl-down` (`CMD-down` in OSX) and the entire row should be selected. At the bottom of the Excel screen you will see the number of elements selected as a quick way to know the number of rows. Repeat the previous procedure and find out how many students are local, HECS or international.

Questions

1. How many students are in your class?

student_list.csv

Home Insert Page Layout Formulas Data Review View

Cut Copy Paste Format

Calibri (Body) 12 A A

B I U

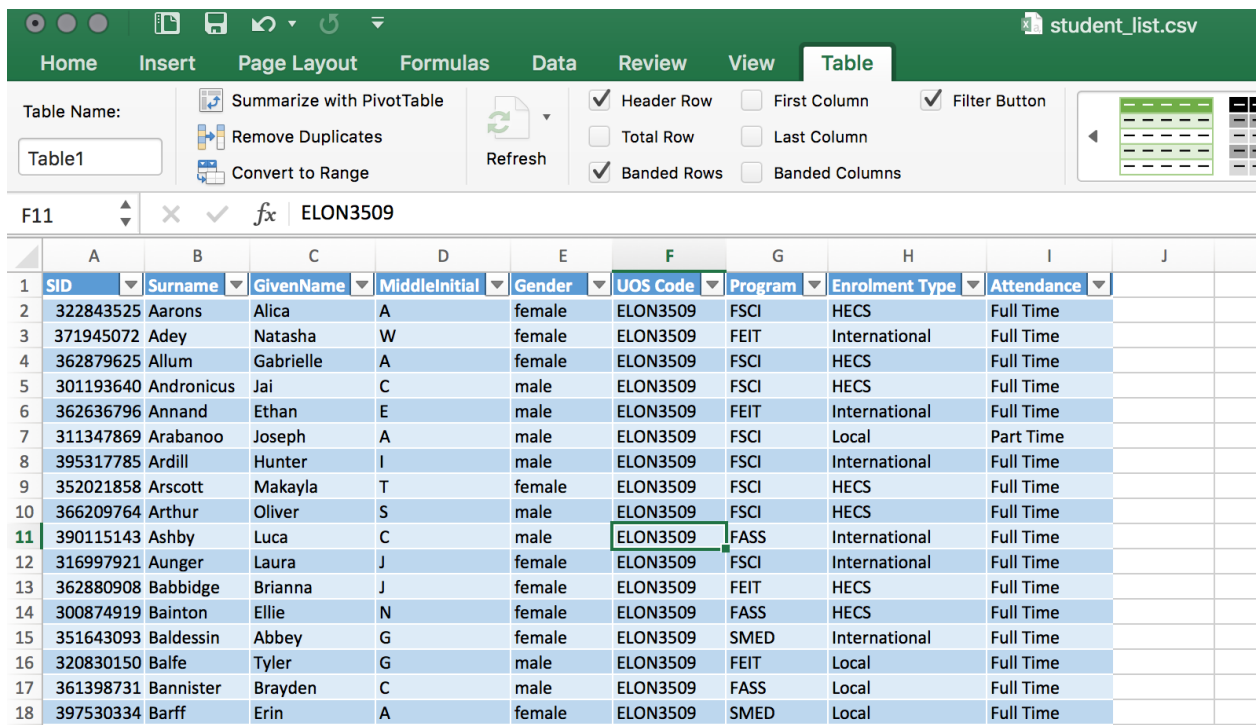
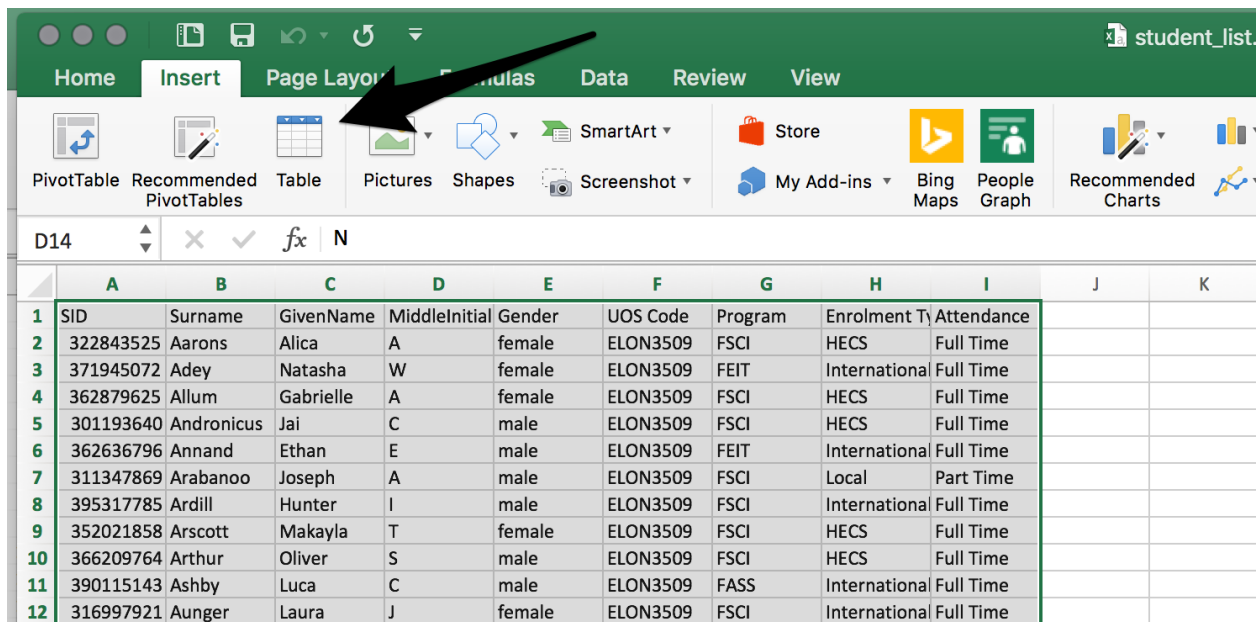
Wrap Text Merge & Center

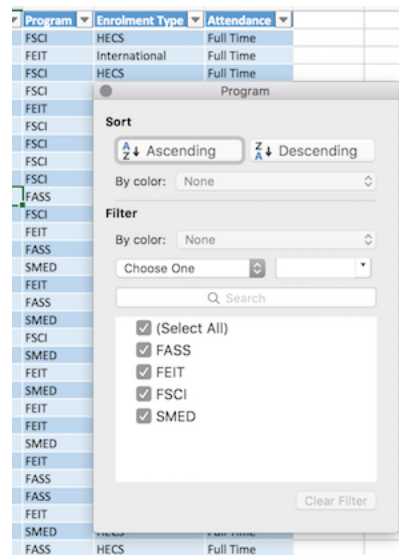
General \$ %

D14 fx N

	A	B	C	D	E	F	G	H	I	J	K	L
1	SID	Surname	GivenName	MiddleInitial	Gender	UOS Code	Program	Enrolment Ty	Attendance			
2	322843525	Aarons	Alica	A	female	ELON3509	FSCI	HECS	Full Time			
3	371945072	Adey	Natasha	W	female	ELON3509	FEIT	International	Full Time			
4	362879625	Allum	Gabrielle	A	female	ELON3509	FSCI	HECS	Full Time			
5	301193640	Andronicus	Jai	C	male	ELON3509	FSCI	HECS	Full Time			
6	362636796	Annand	Ethan	E	male	ELON3509	FEIT	International	Full Time			
7	311347869	Arabanoo	Joseph	A	male	ELON3509	FSCI	Local	Part Time			
8	395317785	Ardill	Hunter	I	male	ELON3509	FSCI	International	Full Time			
9	352021858	Arscott	Makayla	T	female	ELON3509	FSCI	HECS	Full Time			
10	366209764	Arthur	Oliver	S	male	ELON3509	FSCI	HECS	Full Time			
11	390115143	Ashby	Luca	C	male	ELON3509	FASS	International	Full Time			
12	316997921	Aunger	Laura	J	female	ELON3509	FSCI	International	Full Time			
13	362880908	Babbidge	Brianna	J	female	ELON3509	FEIT	HECS	Full Time			
14	300874919	Bainton	Ellie	N	female	ELON3509	FASS	HECS	Full Time			
15	351643093	Baldessin	Abbey	G	female	ELON3509	SMED	International	Full Time			
16	320830150	Balfe	Tyler	G	male	ELON3509	FEIT	Local	Full Time			
17	361398731	Bannister	Brayden	C	male	ELON3509	FASS	Local	Full Time			
18	397530334	Barff	Erin	A	female	ELON3509	SMED	Local	Full Time			
19	355627475	Baylee	Anthony	L	male	ELON3509	FSCI	International	Full Time			
20	395008714	Beach	Tahlia	R	female	ELON3509	SMED	Local	Full Time			
21	352717868	Beeby	Callum	B	male	ELON3509	FEIT	HECS	Full Time			
22	394589352	Belcher	Mackenzie	J	female	ELON3509	SMED	International	Full Time			
23	351106233	Benstead	Caleb	E	male	ELON3509	FEIT	HECS	Full Time			
24	324775672	Blacklow	Brianna	B	female	ELON3509	FEIT	HECS	Part Time			
25	359936485	Bosch	Aidan	B	male	ELON3509	SMED	Local	Full Time			
26	388434168	Bowen	Zara	R	female	ELON3509	FEIT	HECS	Full Time			
27	383530460	Bracy	Stephanie	M	female	ELON3509	FASS	HECS	Full Time			
28	336275645	Bray	Phoebe	C	female	ELON3509	FASS	HECS	Full Time			
29	376434716	Brock	Scarlett	A	female	ELON3509	FEIT	HECS	Full Time			
30	386719553	Buggy	Bella	K	female	ELON3509	SMED	HECS	Full Time			
31	377877923	Bunny	Brayden	B	male	ELON3509	FASS	HECS	Full Time			
32	320834111	Burke	Abby	B	female	ELON3509	FSCI	HECS	Full Time			
33	345171616	Cannan	Lily	J	female	ELON3509	FEIT	International	Full Time			
34	369056671	Challinor	Aaron	J	male	ELON3509	FEIT	International	Part Time			
35	336534413	Chang	Chow	C	female	ELON3509	FEIT	HECS	Full Time			
36	320264344	Chang	Jiao	S	female	ELON3509	SMED	Local	Full Time			
37	386943298	Chang	Lian	M	female	ELON3509	FEIT	International	Full Time			
38	345294497	Chao	Yi Jie	K	female	ELON3509	FEIT	International	Full Time			

student_list +





2. How many *programs* do you have in the data? What is the break out of the students per program?
3. What type of enrollment do you have? What is the percentage of students for each value?
4. What is the gender balance in the course?

1.5.4 Main elements in OnTask

OnTask contains three central concepts:

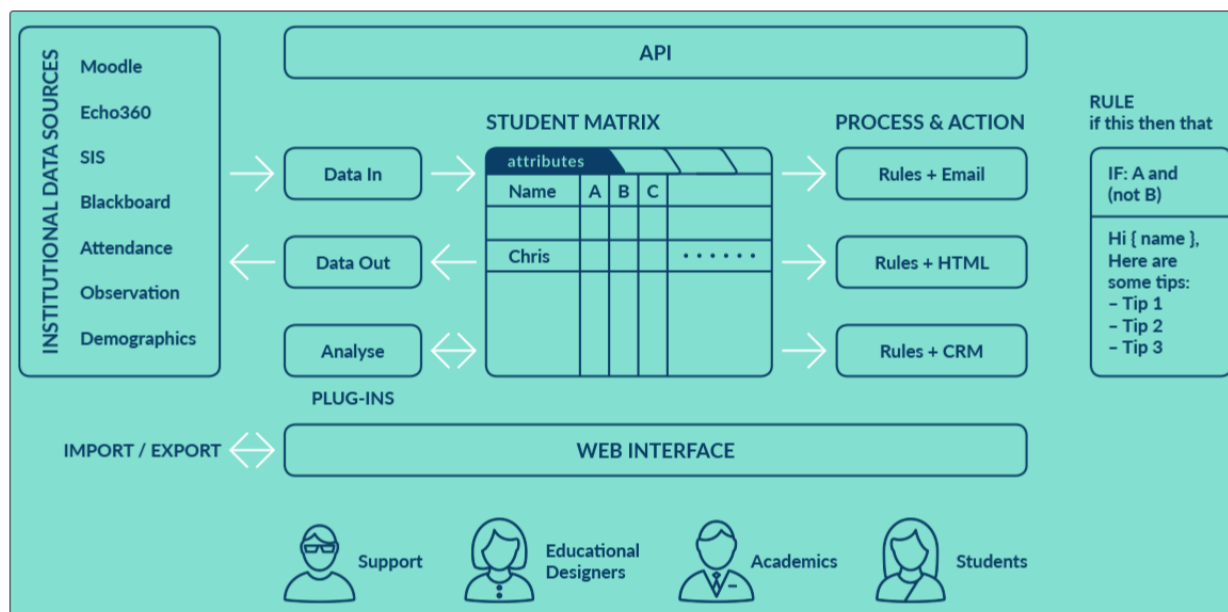
Workflow A workflow is simply an entity that contains a data table and a set of actions. The most intuitive interpretation is to assign a workflow to a *course*. An instructor may participate in various courses. Each of them has a different set of students. A workflow will contain the data about the students in a course, and the actions that are appropriate for that course. After opening a session in OnTask, the user manages a collection of workflows. At this level the platform allows for the creation, modification, deletion, sharing and cloning of workflows. Most of the operations require the user to first *open* the workflow.

Table Each workflow has a data table, a two-dimensional structure in which each row represents a learner, and each column a learner attribute such as the score in an assessment, class attendance, number of interventions in the discussion forum, engagement with videos, etc. The operations over the table allow to create, rename, delete, change the data type, or clone a column, the creation of a table view showing a subset of the data, etc. A table in the workflow must have **at least** one *key column*. Key columns are marked as such in the workflow. This mark can be removed as long as it is not the only remaining key column in the table. Any column can be turned into a key column as long as all its values are non-empty and unique for each row.

Actions An action in OnTask can be:

- A HTML text in which certain parts that are included or excluded based on a set of **conditions** created with the learner attributes (for example, number of interventions in the forum is larger than five, number of times a video was watched is larger than 2, etc). The purpose of this text is to make it available to the learners.
- A set of questions or a survey that is offered to the learners to answer. Upon receiving the responses, the information is incorporated to the workflow table. The survey can also be used by the instructor to capture observations.
- A **JSON object** representing a set of information fields. The values of these fields can be included, excluded or changed depending on a set of **conditions** created with the learner attributes (analogous to those

described for the HTML text). This object can be sent it to an external platform for further processing. The following figure illustrates the high level view of OnTask.



1.5.5 Create an empty workflow

Log into the tool. If you have an instructor account, you will see the *New workflow* button to create a new workflow as shown in the following figure.

The screenshot shows the 'Create a workflow*' page in the OnTask application. At the top, there is a navigation bar with 'Home', 'About', and 'Contact' links. Below the navigation bar, the main heading is 'Create a workflow*'. Underneath, a 'Next Step' instruction reads: 'The main element in OnTask is a *Workflow**. Start by first creating a workflow element using the *New Workflow* button below, and then upload data in its table.' Below this instruction are two buttons: '+ New workflow' and 'Import workflow'. A footnote at the bottom explains that a workflow contains a table with learner data and a collection of actions. The footer shows '© OnTask Project 2019' and 'Version: B.4.0.1'.

Click in the button and enter its name and a description.

After creating the workflow, the next screen has the menu to upload data to the table. The current workflow is shown underneath the top menu as shown in the following figure (the string *BIOL1011* with a blue background).

You can always click in the *Home* link at the top menu to go back to the home page and select a different workflow to open (or create a new one).

Create a workflow

Create a new *empty* workflow (that is, with no data table or actions) by providing a name and a description. The next page will ask you for the data to populate the data table.

Name*

Description text

Cancel

✓ Create workflow

Table

BIOL1011

Data upload/merge operations

Operation	Description
+ CSV Upload/Merge	<p>Upload data contained in a CSV file (typically saved using a spreadsheet or other data manipulation package).</p> <p>If the table is empty, the data will be simply loaded detecting those columns that have unique values for every row.</p> <p>If the workflow table has data, the operation will ask you to match a key column in your table with a key column in the file to upload so that the rows can be matched.</p>
+ Excel Upload/Merge	<p>Upload data from Excel file.</p> <p>Given a file name and the name of a sheet, upload or merge the content of the Excel file.</p> <p>If the workflow table has data, the operation will ask you to match a key column in your table with a key column in the file to upload so that the rows can be matched.</p>
+ Google Sheet Upload/Merge	<p>Upload data from a Google Sheet file.</p> <p>Given the URL pointing to a publicly available Google Sheet upload or merge its content.</p> <p>If the workflow table has data, the operation will ask you to match a key column in your table with a key column in the sheet so that the rows can be matched.</p>
+ SQL Connection	<p>Upload data stored in a remote database by defining connections.</p>

© OnTask Project 2019

Version: B.4.0.1

Open or create a workflow*

+ New workflow Import workflow

BIOL1011

Course on Cell Biology

14 rows, 67 columns, 6 actions.

Open Edit Clone Delete

*A workflow contains a *table* with data about a set of learners and a *collection of actions*. The table is organised by *columns* and the information in each row corresponds to one learner. The actions will behave differently depending on the values stored for every learner and a set of conditions. These conditions can be written based on the values of the columns for each student. Typically, you create a workflow for a cohort of learners for which you have data available. Then you write the actions depending on a) when you want students to introduce data or b) when you want students to receive personalised data. The personalisation is done writing conditions to enable/disable text or other elements in a page. The page can either be sent to the learner through email, shown through a HTML address, or sent to another machine.

© OnTask Project 2019 Version: B.4.0.1

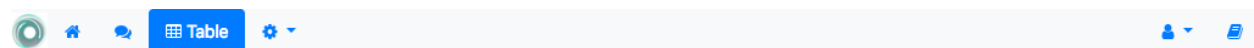
Each workflow is represented by an element showing its name, description, size of the table, and number of actions. Additionally, the icons in the bottom of the element allow the following operations:

- *Open* the workflow for manipulation
- *Rename* and change the workflow description,
- *Clone* the workflow (create an exact replica with a modified name),
- *Delete data* in the workflow table, or
- *Delete* the workflow.

From any screen in the platform, the icon in the top right corner next to your profile image is a link to the OnTask documentation.

1.5.6 The Top Menu

OnTask assumes that you will be working with one workflow. The first screen (the one that appears when you log in or when you click in the *Workflows* link in the top menu) shows you the workflows available. To start working with one of them you have to *Open* it. Once you open a workflow, the top menu appears as shown in the following figure.



Superuser

If your account has *superuser* or *admin* privileges, the top menu will include the additional option with name *SQL Connections*. This functionality allows the administrator to manage a collection of connections to SQL relational databases. These connections are available to instructors as part of the operations to Upload/Merge data into the table.

The menu contains the following links:

Home Link back to the home page showing the workflows available for manipulation.

Actions Create, edit and execute actions.

Table Operations to visualize and manipulate the table (search for values, add a row, add a column)

Settings Additional operations to view:

Workflow operations This page contains links to export, rename, clone, flush the data, or delete the workflow. It also allows you to manipulate a set of attributes, and add users to share the workflow.

Column operations This page shows information about the columns in the workflow.

Scheduled actions Shows which actions have been scheduled for execution.

View logs A table showing the history of operations performed on this workflow

User profile Link to the user profile and logout

Manual Link to the OnTask manual (this document)

1.5.7 Data Upload

Log into the platform and:

- Create a new workflow, introduce name and description or
- Open an existing workflow without data, click in the button with name *Manage table data* and then in the *Data Upload/Merge* link.

The following page allows you to upload data from three sources:

- a CSV (comma separated values) file,
- an Excel file,
- a publicly available Google Sheet, or
- a SQL connection to a relational database.

Click in the *CSV Upload/Merge* button. The next screen asks you to choose a file to upload the data. A CSV file is a text file in which the data is organized by lines, and the data in each line is separated by commas. A conventional spreadsheet program can save the data in this format. When uploading the file you can optionally specify a number of lines to skip at the top or bottom of your data file. This is useful when the CSV file is produced by another tool and contains some of these lines that have to be ignored.

Choose the file `all_data.csv` and click in the *Next* button. The next screen shows the name of the columns detected in the file, the type of data in the column (also automatically detected), a pre-filled field with the column name (in case you want to change it), and if it is a *key column* (there are no repeated values in all the rows).






The *key columns* are highlighted because a workflow must have at least one column of this type. Select all the columns (clicking in the top element labeled *load*) and click on the *Finish* button. The data in the file has now been uploaded to the table of the selected workflow.

1.5.8 Browsing the data table



Log into the platform and open a workflow that has data in the table. Click in the *Table* link at the top of the screen to see the stored data:

The buttons at the top of the page offer the following functions:

Row A form to introduce a new row in the table with one field per column.






BIOL1011



Data upload/merge operations

Operation	Description
+ CSV Upload/Merge	<p>Upload data contained in a CSV file (typically saved using a spreadsheet or other data manipulation package).</p> <p>If the table is empty, the data will be simply loaded detecting those columns that have unique values for every row.</p> <p>If the workflow table has data, the operation will ask you to match a key column in your table with a key column in the file to upload so that the rows can be matched.</p>
+ Excel Upload/Merge	<p>Upload data from Excel file.</p> <p>Given a file name and the name of a sheet, upload or merge the content of the Excel file.</p> <p>If the workflow table has data, the operation will ask you to match a key column in your table with a key column in the file to upload so that the rows can be matched.</p>
+ Google Sheet Upload/Merge	<p>Upload data from a Google Sheet file.</p> <p>Given the URL pointing to a publicly available Google Sheet upload or merge its content.</p> <p>If the workflow table has data, the operation will ask you to match a key column in your table with a key column in the sheet so that the rows can be matched.</p>
+ SQL Connection	<p>Upload data stored in a remote database by defining connections.</p>

© OnTask Project 2019
 Version: B.4.0.1

BIOL1011

Step 1: Select CSV file

all_data.csv

File in CSV format (typically produced by a statistics package or Excel)




Lines to skip at the top

Number of lines to skip at the top when reading the file


Lines to skip at the bottom

Number of lines to skip at the bottom when reading the file



© OnTask Project 2019
 Version: B.4.0.1

Table



BIOL1011




Step 2: Select Columns

Cancel


Back

> Finish



Load <input checked="" type="checkbox"/>	Name	Rename to	Type	Key
<input checked="" type="checkbox"/>	SID	<input type="text" value="SID"/>	number	Keep? <input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Identifier	<input type="text" value="Identifier"/>	string	Keep? <input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	email	<input type="text" value="email"/>	string	Keep? <input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Surname	<input type="text" value="Surname"/>	string	Keep? <input type="checkbox"/>
<input checked="" type="checkbox"/>	GivenName	<input type="text" value="GivenName"/>	string	Keep? <input type="checkbox"/>
<input checked="" type="checkbox"/>	MiddleInitial	<input type="text" value="MiddleInitial"/>	string	
<input checked="" type="checkbox"/>	Full name	<input type="text" value="Full name"/>	string	Keep? <input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Gender	<input type="text" value="Gender"/>	string	
<input checked="" type="checkbox"/>	Course Code	<input type="text" value="Course Code"/>	string	
<input checked="" type="checkbox"/>	Program	<input type="text" value="Program"/>	string	
<input checked="" type="checkbox"/>	Enrolment Type	<input type="text" value="Enrolment Type"/>	string	
<input checked="" type="checkbox"/>	Attendance	<input type="text" value="Attendance"/>	string	
<input checked="" type="checkbox"/>	Q01	<input type="text" value="Q01"/>	number	
<input checked="" type="checkbox"/>	Q02	<input type="text" value="Q02"/>	number	
<input checked="" type="checkbox"/>	Q03	<input type="text" value="Q03"/>	number	

Table



BIOL1011

Table

Next Step: Click in the [Actions](#) menu at the top bar to create actions to execute with the data in the table.

+ Row

+ Column ▾































Manage table data ▾

Views

Dashboard

CSV Download

Show entries Search:

	SID ↑↓	Identifier ↑↓	email ↑↓	Surname ↑↓	GivenName ↑↓	MiddleInitial ↑↓	Full name ↑↓	G
  	300773633	Participant 8	euhz6532@bogus.com	Tung	Mei	S	Mei Tung	
  	315500979	Participant 11	zqdk2609@bogus.com	Jen	You	Y	You Jen	
  	317547296	Participant 9	ckrn7263@bogus.com	Chao	Yuan	Q	Yuan Chao	
  	319549896	Participant 2	ttqp9766@bogus.com	Meng	Huan Yue	F	Huan Yue Meng	
  	320250819	Participant 14	tjxk6150@bogus.com	Shen	Hu	L	Hu Shen	
  	326285587	Participant 6	isim6886@bogus.com	Hou	Lian	S	Lian Hou	
  	330794979	Participant 12	olie8242@bogus.com	Arthur	Oliver	S	Oliver Arthur	
  	333306652	Participant 1	qvr9413@bogus.com	Tuan	Jian	A	Jian Tuan	
  	352120481	Participant 4	ucdx8510@bogus.com	Mueller	Matthew	G	Matthew Mueller	
  	360504510	Participant	zckx0360@bogus.com	Huan	Zachary	G	Zachary	

Column Add a new column to the table. The pull-down menu also allows to add a column with values calculated with a formula and using the existing values, or a column with random values (See [Adding a column](#) for a detailed explanation).

Manage table data Menu to upload/merge data to the table (see [Data](#) for a detailed explanation), execute a plugin, or delete the data in the table.

Views A view is a screen showing a subset of rows and/or columns in the table. Clicking in the Views link will take you to another screen to [manage these views](#). The pull down menu offers links to access the subset of the table defined by the available views.

Dashboard The dashboard shows a statistical summary for the columns shown in the table. This number may be too high, so you should consider using [Table Views](#) to reduce the amount of information shown in the dashboard.




CSV Download This functionality downloads the data in the table as a CSV file. Combine this functionality with the [Table Views](#) to handle large tables.

Table Views


For tables with a large number of columns and/or rows OnTask allows you to define a *view* of the table that shows only a subset of it. To create a view click first in the *Views* button at the top of the table page and then the *Add View* in the next page. Insert a name, description and select some of the columns as shown in the following figure.

You can also define views to show only a subset of rows. The subset is selected using a *row filter* stating the conditions that must be satisfied by a row to be included in the view. These conditions are stated in terms of the column values.

Save the view and then click in the *View* button. The appropriate table subset is shown. The buttons at the top of the page allow you to edit the view (change the rows and columns selected), or select another available view.

Table



BIOL1011

Subset 1

Next Step: Click in the [Actions](#) menu at the top bar to create actions to execute with the data in the table.

+ Row

+ Column

Manage table data

Views

Edit view

Full table

Dashboard































CSV Download

Show

10

 entries

Search:

	email	Program	Enrolment Type
  	ckrn7263@bogus.com	FEIT	International
  	ctfh9946@bogus.com	FASS	HECS
  	euho9752@bogus.com	SMED	Local
  	euhs6532@bogus.com	FSCI	HECS
  	isim6886@bogus.com	SMED	International
  	kjbc6748@bogus.com	FSCI	HECS
  	olie8242@bogus.com	FSCI	International
  	pzaz8370@bogus.com	FEIT	HECS
  	qvr9413@bogus.com	FASS	HECS
  	tcnf7608@bogus.com	FASS	HECS

Showing 1 to 10 of 15 entries

Previous

1

2

Next

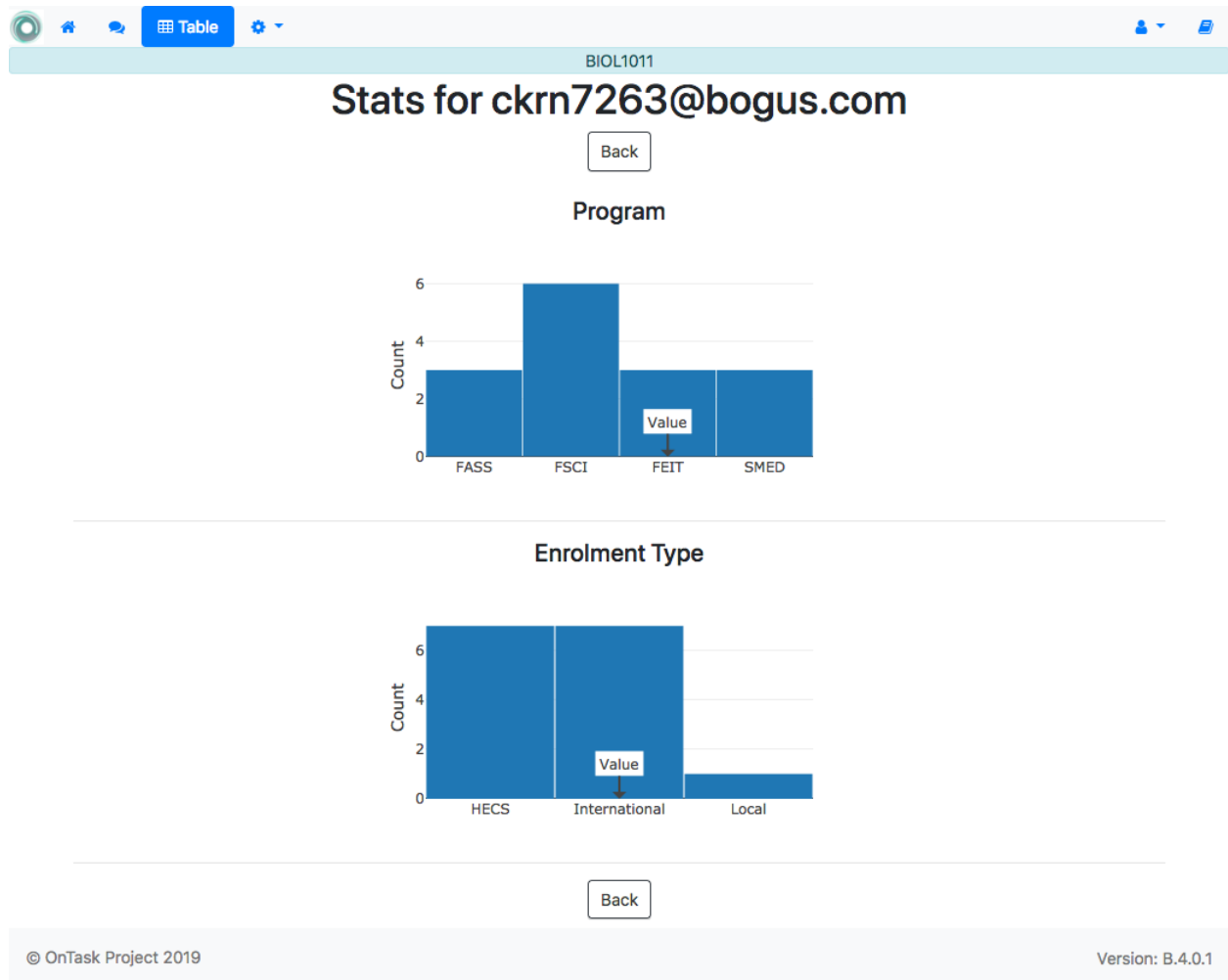
You may drag-and-drop a column header to change its position in the table

© OnTask Project 2019

Version: B.4.0.1

Column and Row Statistics

If you click in one of the *Statistics* icons in the left-most column of a row OnTask shows a page with statistical representation of the population values for the columns and a mark denoting where is the value of the selected row.



1.5.9 Actions

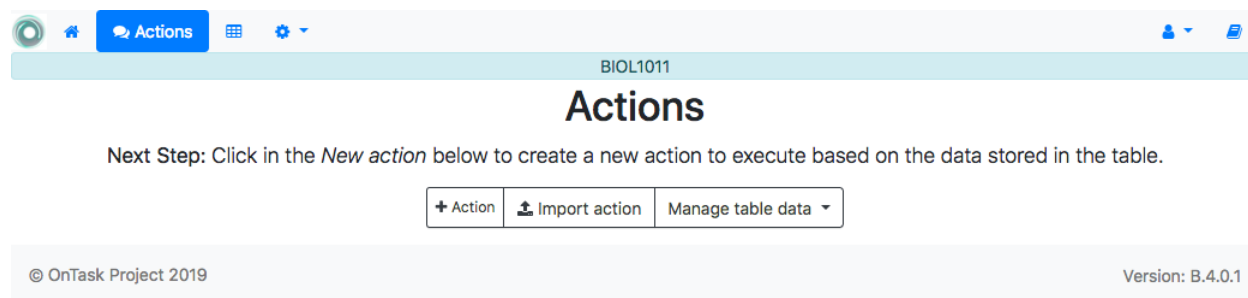
Log into OnTask and open a workflow. Access the *Actions* page clicking in the link with the same name in the top of the screen. The next screen shows the list of actions that are part of the workflow, and the buttons to create a new one, import, or manage the data table.

OnTask contains functions to manage three type of actions:

Personalized text These actions allow to create a **HTML text** in which some of its elements may be present/ignored depending on the how a condition is evaluated for every student.

Surveys A set of questions for the students to answer. The information is collected and automatically added to the table in the workflow.

Personalized JSON object These actions allow to create a **JSON object** with elements in its structure that can be present or ignored depending on how a condition is evaluated for every student (similarly to the previous actions)



1.5.10 Personalized text action

Click on the button to create a *New Action*, provide a name, a description (optional) and select the type *Personalized text*.

The next screen is the *personalized text action editor*. The functions in this page are divided into three areas represented by the tabs: Personalized Text, Filter Learners and Text Conditions. The following figure shows an example of this page.

Personalized Text This area is the HTML text editor to write the content to personalize. It contains pull-down menus to insert column values and attributes.

Filter Learners This section allows to define a *filter*, or a condition to select a subset of the learners for which this action will be considered.

Text Conditions This section contains the conditions to be used in the personalized text.

Place the cursor in the text area and start the text with a salutation. Then click in the pull down menu next to *Insert Column Value*. Select the column *GivenName*. The string `{{ GivenName }}` appears in the text area. This notation is to instruct the next steps to replace the value among double curly braces with the name of each student.

Select the *Text Conditions* tab and click in the *Condition*. A form appears to introduce the name, description and a formula. The formula may contain any combination of Boolean operators connecting column names. For example, the condition:

The screenshot shows the 'Program advice' editor in the OnTask application. The interface includes a top navigation bar with 'Actions', a grid icon, a settings icon, and a user profile icon. Below this is a header bar with 'BIOL1011'. The main title is 'Program advice'. There are two tabs: 'Personalized Text' (active) and 'Text Conditions (0)'. A 'Filter Learners (Empty)' link is also present. A dropdown menu shows '- Insert Column Value -'. The editor area has a rich text toolbar with icons for bold, italic, underline, strikethrough, link, unlink, bulleted list, numbered list, indent, outdent, text color, background color, table, link, unlink, and a code icon. The text area is empty. At the bottom of the editor are 'Preview', 'Save', and 'Close' buttons. The footer shows '© OnTask Project 2019' and 'Version: B.4.0.1'.

Personalized Text

Filter Learners (Empty)

Text Conditions (0)

- Insert Column Value -

© OnTask Project 2019

Version: B.4.0.1

This close-up view of the editor shows the text 'Dear {{ GivenName }}' entered in the text area. The toolbar and dropdown menu are visible above the text area.

- Insert Column Value -

Dear {{ GivenName }}

```
Program equal to FASS
```

can be encoded in the formula as shown in the following figure

We now are going to use this condition to control the appearance of text in the editor. You need to perform the following three steps:

- Switch to the Personalized Text editor tab.
- Write a sentence in the editor with a message that should apply only to those students in the FASS program.
- Select it with the mouse.
- Click in the pull down menu with the text *Use condition in highlighted text*

The result of these steps is shown in the following figure

The text message you wrote for those students in the FASS program is now surrounded by two marks:

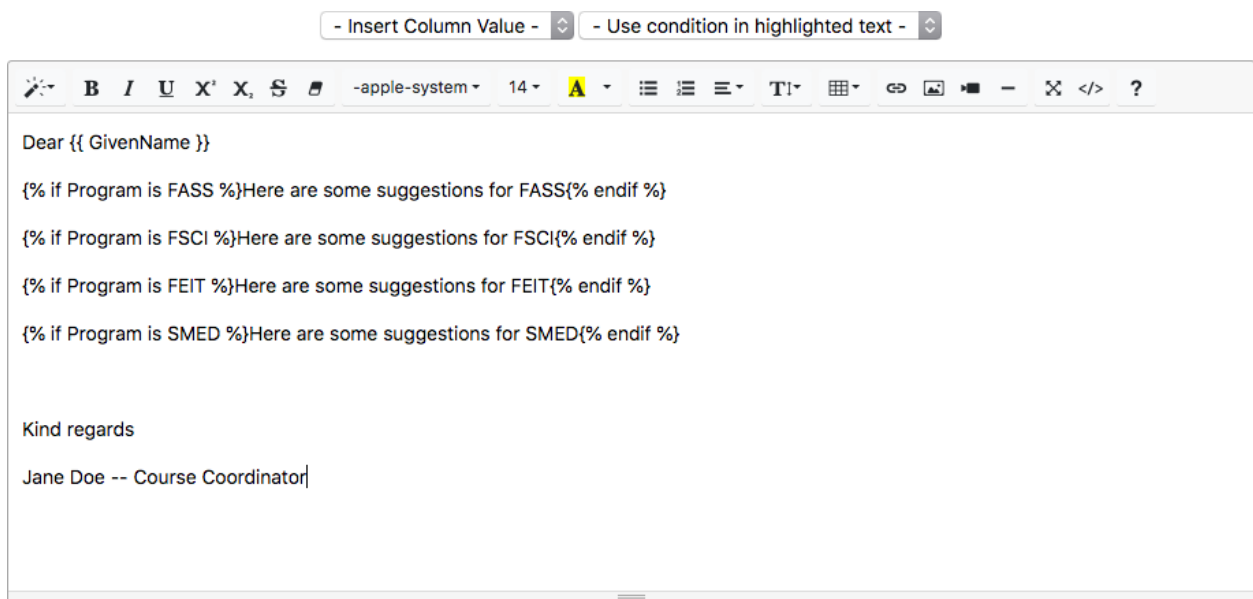
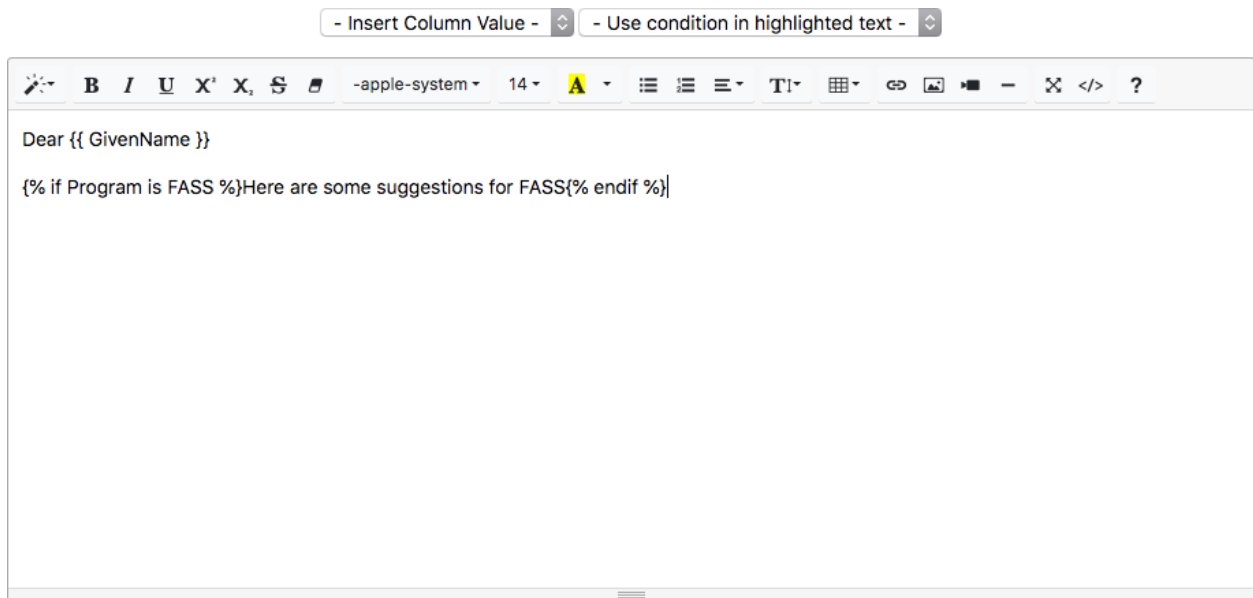
```
{% if Program is FASS %}Here are some suggestions for FASS{% endif %}
```

This is the format to instruct OnTask to check the value of the condition `Program is FASS` and include the surrounded text only if the condition is true.

Repeat the procedure and create three more conditions such that they are true if the value of the *Program* columns is equal to *FSCI*, *FEIT*, and *SMED* respectively. Insert three more messages in the text area that are controlled by their respective conditions. At the end, each sentence will be surrounded by text referring to each of the four conditions you created:

Selecting only a subset of learners

In some cases, you may want to create a personalized text only for a subset of the learners. This can be done defining a *Filter* at the top of the screen (area 1). For example, the following filter:

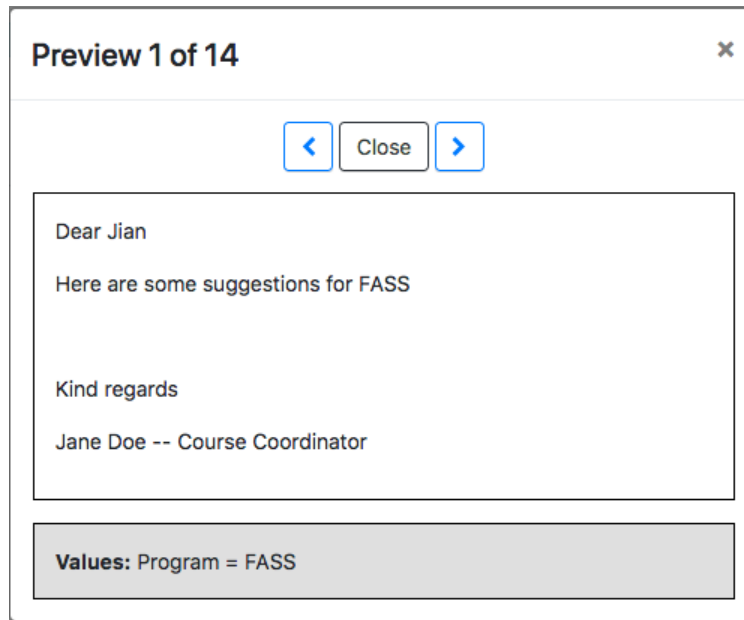


selects only those learners for which the column *Attendance* in the table has the value *Full Time*. Upon closing the small window with the filter data, the editor screen contains information about how many learners are being selected by that filter.

The application will show a message if the filter excludes all the learners (none of them satisfy the condition given in the filter).

The text in the editor will be processed for each row in the table (or the subset specified by the filter) and all conditions and fields will be evaluated with the values in that row. In other words, if a sentence is surrounded by one condition, the condition is evaluated replacing the column names by the values in the row. If the condition is *True*, the text will appear, and if not, it will be ignored.

The button with name *Preview* at the bottom of the editor allows you to see the text resulting from the evaluation of each row.



The bottom part of the screen shows the values of those columns that were used to evaluate the conditions required in the text. You may use the arrows at the top of the screen to review the message and see how it is changing from learner to learner.

1.5.11 Sending Personalized Text through Email

Log into the platform, open a workflow that has data and actions already defined and click the *Action* link in the top menu.

After creating a personalized text action, you may want to send emails to the learners (or a subset of them if you *defined a filter* in the action). Click in the button with name *Email* in the operations of the personalized text action. The following form appears in the screen:

The form allows you to specify the following fields:

Email subject String to use as subject for all the emails.

Column to use as email address This is the name of the column in the table from where to extract the email address to use in the *To* field of the email.

Comma separated list of CC emails This is useful when you want to send the emails with copy to other users. The CC emails must be separated by commas (e.g. *user1@bogus.com, user2@bogus.com*)

Comma separated list of BCC emails Field analogous to the previous one except that the values are used in the blind copy of the email.

Check/exclude email addresses before sending If selected the platform will offer you a last chance to specify some email addresses to *exclude* from the emails. This may be useful if you want to remove a small amount of addresses that you know they should not be considered but they cannot be easily removed with the use of the action filter.

Send you a summary message? If selected the platform will send you (the email you used to log in) a message with a summary of the operation once all emails are sent.

Track email reading in an extra column? If selected the platform will insert an extra column in the table containing the number of times each email has been opened (this counter, though, may not have a correct value as it depends on the configuration of external programs)

BIOL1011

Send emails for action "Program advice"

14 Emails will be sent.

Email subject*

Connecting your program with this course

Column to use for target email address*

email

Comma separated list of CC emails

tutor1@bogus.com, tutor2@bogus.com

Comma separated list of BCC emails

coursecoordinator@bogus.com

☒ Check/exclude email addresses before sending?

☐ Send you a summary message?

☐ Track email reading in an extra column?

☐ Download a snapshot of the workflow?

A zip file useful to review the emails sent.

Cancel Preview > Next

© OnTask Project 2019 Version: B.4.0.1

Download a snapshot of the workflow When selected, the platform saves the workflow in its current state. This function is useful to keep an exact replica of the state of the actions, conditions and data when the data was sent. The resulting file can then be imported (see the *Import workflow* in the home page) to check the content of the emails.

Introduce a string in the subject field and click in the button *Preview* at the bottom of the page.

The new page shows the final appearance of the message with its subject and the values for those columns used to configure the text. Click in the arrows next to the *Close* button to traverse the various messages and verify that they are properly composed.

Final look before pushing the GO button

When running an action sometimes you may want to verify that the list of rows to be processed is correct, or perhaps in the last step remove one or two rows that you know should not be included (and you didn't filter with the action filter). If you want to go through this extra step *before* running the action, tick the field to *Check/exclude* these items. Click in the *Next* button. In the following page you can review the list of items and mark those you want to exclude.

1.5.12 Making content available through OnTask

An alternative to send an email is to make the personalized text available through a URL that is provided by OnTask. This URL can be enabled by clicking in the *URL* button of the action operations.

If learners are allowed to connect to OnTask (they have an account), their email is part of the data stored in the workflow, and the access to the platform can be done through a Single Sign-on system (e.g. LTI through a learning

Preview 1 of 14 ✕

< Close >

Subject: Connecting your program with this course

Dear Jian

Here are some suggestions for FASS

Kind regards

Jane Doe -- Course Coordinator

Values: Program = FASS

Actions BIOL1011

Select items to exclude from action

Values to exclude

Select all Select none

- ☒ qvrr9413@bogus.com
- ☒ ttqp9766@bogus.com
- ☒ pzaz8370@bogus.com
- ☐ ucdx8510@bogus.com
- ☐ tdrv2640@bogus.com
- ☐ isim6886@bogus.com
- ☐ kjbc6748@bogus.com
- ☐ euhz6532@bogus.com
- ☐ ckrrn7263@bogus.com

x qvrr9413@bogus.com x ttqp9766@bogus.com x pzaz8370@bogus.com

Click here to search

Cancel Back Send

© OnTask Project 2019 Version: B.4.0.1

URL to the personalised content

✕

This URL provides access to the content personalised for each user.

`http://localhost:65517/action/1/serve/`

☐ URL available to users?

Action available from

Action available until

Access is protected by authentication, and the user **email** is in a column named **email**. To use a different column, add this parameter to the end of the URL:

`?uatn=[COLUMN NAME]`

replacing `[COLUMN NAME]` by the column to use instead of **email**.

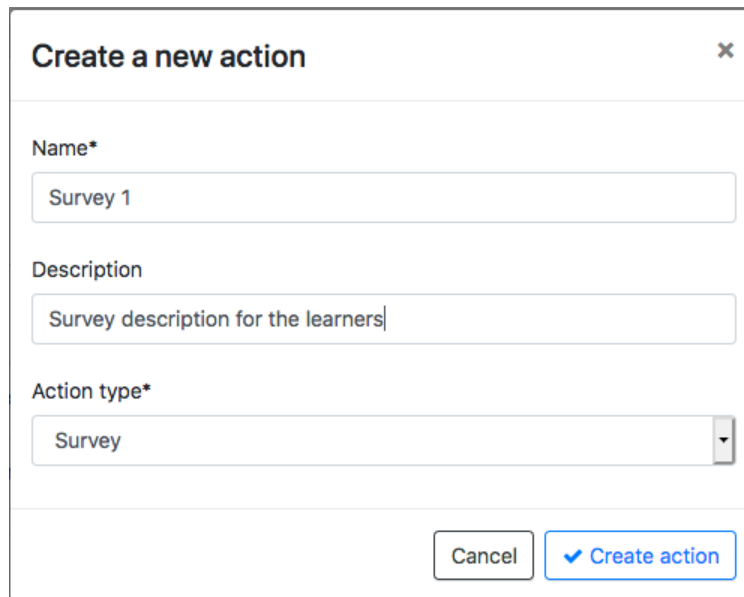
✓ Ok

management system), the URL shown in the previous figure will allow them to access the personalized text.

1.5.13 Surveys and/or Polls

Personalized texts can be very powerful to offer students content, comments, or point them to resources selected based on the available data. This data may come from sources such as student information systems or the learning management system. However, in some cases we would like to obtain data either directly from the students, or perhaps from the instructors through observation. This functionality is offered in OnTask through the actions called *Surveys*. These surveys collect information from the students (or the instructors) through conventional web forms and the data is automatically added to the workflow table and available to be explored through the dashboard or statistics.

The first step to create a survey is to go to the *Actions* page, click in the *New action* button, provide a name, a description (optional) and select the action type *Survey*:



After the survey is created, the following screen is shown

The editor contains three tabs:

Survey Questions The columns used to collect the data. In this action, a column is equivalent to a question. The description of the column is the text of the question. In this part of the editor you may either use one of the existing columns as question, create a new column (or question), or create a new derived one (the initial values are created by combining values from other columns).

Filter learners An expression identical to the one used in the personalized text to select a subset of the learners for which the will be available.

Survey Parameters This tab contains additional parameters for the survey, more precisely:

Description Text describing the survey that is shown to the learners.

Key column to identify learners The key column in the table that will be used to identify the users when submitting their answers. This is typically the column that contains the user email.

Shuffle questions If selected, the questions in the survey will be shuffled when shown to the learners.

This page will show you a warning message if any of the columns used in the survey has an empty description. Let's suppose you want to ask the learners two questions:

- What was the most challenging topic for you this week?

BIOL1011

Survey 1

Survey Questions Filter Learners (Empty) Survey parameters

There must be at least one non-key column.

- Insert existing question - + Create question

Show 10 entries Search:

Name	Description (shown to learners)
No data available in table	

Showing 0 to 0 of 0 entries

Previous Next

Preview Done

© OnTask Project 2019 Version: B.4.0.1

- What was your dedication to the course this week?

To make the data suitable for further processing, we will create the two questions/columns with a set of pre-defined answers. Use the *Create question* button to create two columns of type string and provide the allowed answer values as a comma-separated list.

Repeat the procedure for the second question/column. The result should be as shown in the following figure.

As in the case of the personalized text action, the *Preview* button allows you to verify how will the learners see the content:

Once created, you may select the URL from the action as it was described for the personalized text and make it available for learners to enter their answers. The data will be automatically added to the table.

The *Run* button in the survey actions allows an instructor to introduce the survey answers for multiple learners. This functionality is used for instructors to enter observations when interacting with learners. The table search functionality allows to find the learners quickly and then click in their survey execution and enter the data.

1.5.14 Run a survey

Once a survey has been created, there are two possible operations that it supports:

- 1) Provide a URL for those learners for which there is a row in the data table to introduce the requested information. This operation is achieved by simply clicking in the URL operation in the action table. The next screen will show the URL and the possibility to enable/disable it and even define a time window during which the URL is valid. The following figure shows an example of this screen
- 2) The instructor selects a learner and introduces the data for the questions. This function is accessed clicking in the *Run* operation in the action table. The application shows the table of learners (with links to modify their data) together with the data collected so far. The following figure shows an example of this screen

Add question ✕

Question name*

Description (shown to the learners)

Type of data to store in the column*

string ▼

Question position (zero to insert last)*

0 ⬆ ⬇ ⬆

Question active from

📅

Question active until

📅

Comma separated list of values allowed in this column

Cancel

✓ Add question

BIOL1011

Survey 1

Survey Questions [Filter Learners \(Empty\)](#) [Survey parameters](#)

- Insert existing question - [+ Create question](#)

Show 10 entries Search:

Name	Description (shown to learners)	
Survey Q1	What was the most challenging topic for you this week?	
Survey Q2	What was your dedication to the course this week?	

Showing 1 to 2 of 2 entries

[Previous](#) **1** [Next](#)

[Preview](#) [Done](#)

© OnTask Project 2019 Version: B.4.0.1

Preview 1 of 15

[Close](#)

Survey description for the learners

What was the most challenging topic for you this week?

What was your dedication to the course this week?

URL to the personalised content ✕

This URL provides access to the content personalised for each user.

`http://localhost:64709/action/26/serve/`

☐ URL available to users?

Action available from






Action available until

Access is protected by authentication, and the user **email** is in a column named **email**. To use a different column, add this parameter to the end of the URL:




`?uatn=[COLUMN NAME]`

replacing `[COLUMN NAME]` by the column to use instead of **email**.

✓ Ok






BIOL1011




Student comments Week 1





Feedback about today's lecture

Show entries

email 	Difficult Week 1 	Sat W1 
ckrn7263@bogus.com		
ctfh9946@bogus.com		
euho9752@bogus.com		
euhz6532@bogus.com		
isim6886@bogus.com		
kjbc6748@bogus.com		
olie8242@bogus.com		
pzaz8370@bogus.com		
tcnf7608@bogus.com		
tdrv2640@bogus.com		

Showing 1 to 10 of 14 entries



© OnTask Project 2019

Version: B.4.0.1

1.5.15 Combining personalized text and surveys

The information collected through `:ref:surveys<tut_surveys>` is stored in the appropriate columns in the data table and therefore can be used in a personalized text action to select the appropriate message. For example, the information collected as answers to the question *What was the most challenging topic for you this week?* can be used to select a set of appropriate links to resources about the given topic. The answers are stored in column *Survey Q1* in the Table. Analogously, the answers to the question *What was your dedication to the course this week?* are stored in the Table in the column with name *Survey Q2*.

The values in these columns can be used to create a personalized text action that provides a text with suggestions for additional resources or techniques to adopt based, for example, on the program in which the student is enrolled and the topic they found more challenging.

When using the values previously collected in a survey, special care must be taken to account for those learners that did not answer the survey (or did so partially). This can be easily achieved by adding additional conditions or simply filtering those students for which any of the columns in the survey is null (*Survey Q1 is null*).

The screenshot shows a modal window titled "Edit filter" with a close button (X) in the top right corner. Inside the window, there is a "Description" section with a text input field containing "Complete data". Below this is a section titled "The learner will be selected if". This section contains a logic builder with three buttons: "NOT", "AND", and "OR", where "AND" is currently selected. To the right of these buttons are two green buttons: "+ Add rule" and "Add group". Below the logic buttons, there are two rules listed. The first rule is "Survey Q1" followed by a dropdown menu showing "is not null" and a red "Delete" button. The second rule is "Total" followed by a dropdown menu showing "is not null" and a red "Delete" button. At the bottom of the dialog, there are two buttons: "Cancel" and "Update".

The action then uses the conditions to control which message is shown.

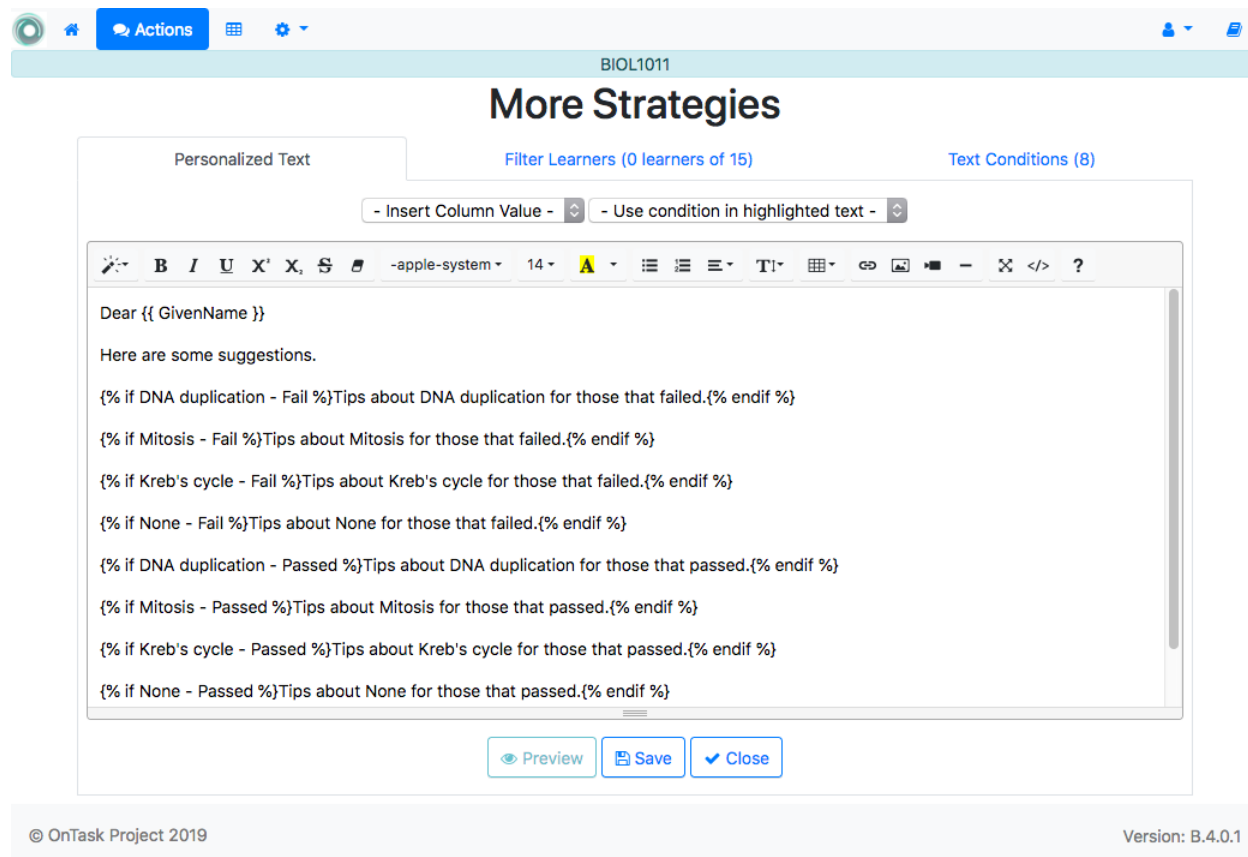
1.5.16 Import a pre-defined survey

OnTask offers a [library of pre-defined surveys](#) that can be uploaded directly into one of your existing workflows. Go to the *Action* page of an existing workflow (or alternatively create a workflow and upload some data). Download one of the [pre-defined tests](#) and store the file in your computer.

Click in the *import action* button.

Enter a name for the survey and select the file stored in your computer. Click in the *Import* button.

Edit the action and review the questions, filter, and survey parameters. Make sure you selected the *key column* that is going to be used to identify the users.



BIOL1011

More Strategies

Personalized Text Filter Learners (0 learners of 15) Text Conditions (8)

- Insert Column Value - - Use condition in highlighted text -

Dear {{ GivenName }}

Here are some suggestions.

{% if DNA duplication - Fail %}Tips about DNA duplication for those that failed.{% endif %}

{% if Mitosis - Fail %}Tips about Mitosis for those that failed.{% endif %}

{% if Kreb's cycle - Fail %}Tips about Kreb's cycle for those that failed.{% endif %}

{% if None - Fail %}Tips about None for those that failed.{% endif %}

{% if DNA duplication - Passed %}Tips about DNA duplication for those that passed.{% endif %}

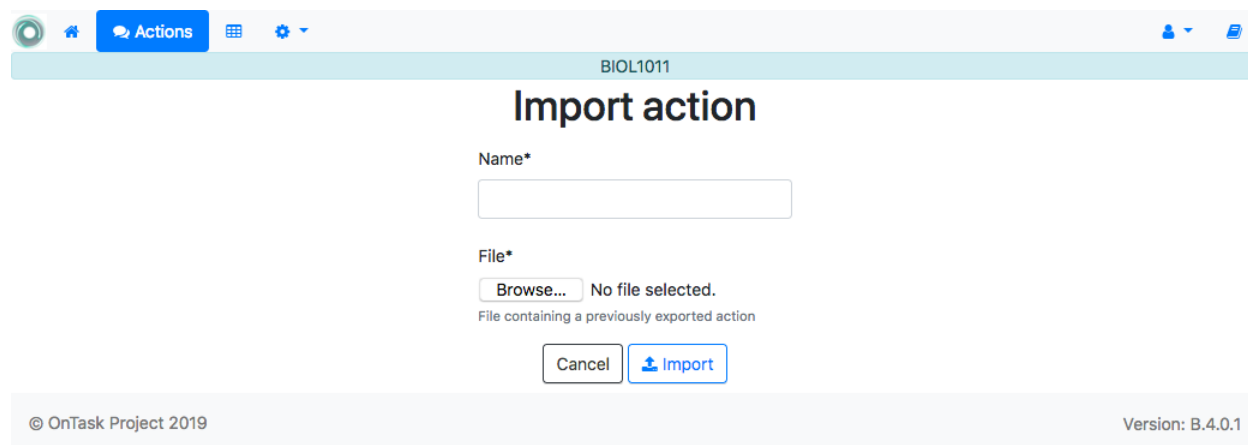
{% if Mitosis - Passed %}Tips about Mitosis for those that passed.{% endif %}

{% if Kreb's cycle - Passed %}Tips about Kreb's cycle for those that passed.{% endif %}

{% if None - Passed %}Tips about None for those that passed.{% endif %}

[Preview](#) [Save](#) [Close](#)

© OnTask Project 2019 Version: B.4.0.1



BIOL1011

Import action

Name*

File*

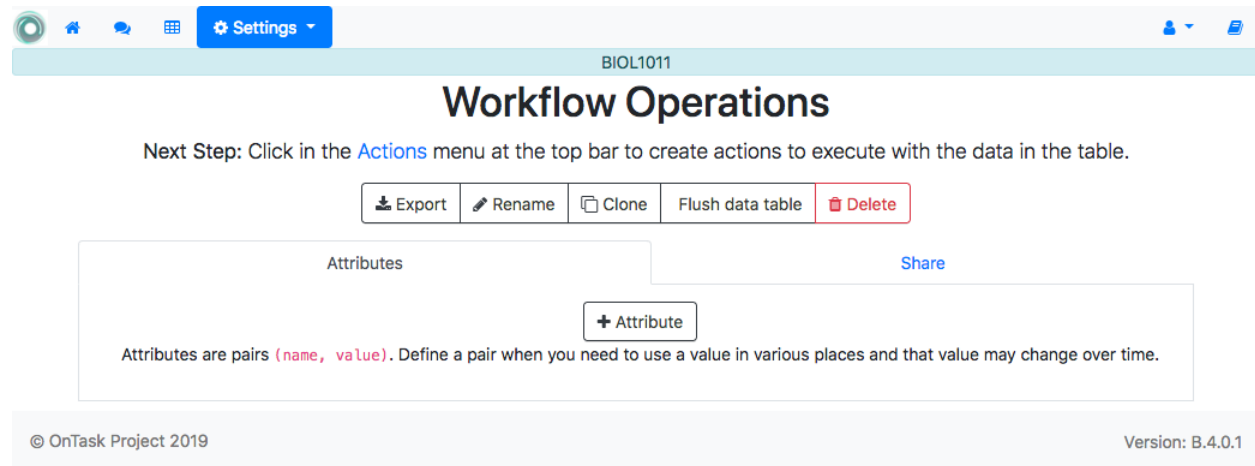
[Browse...](#) No file selected.
File containing a previously exported action

[Cancel](#) [Import](#)

© OnTask Project 2019 Version: B.4.0.1

1.5.17 Workflow operations

The link *Workflow operations* available in the pull down menu of the *Settings* link in the top bar menu leads to a page that offers information and several additional operations as shown in the following figure.



The links at the top of the page offer the operations to:

Export This functionality allows you to take all the information included in a workflow and export it. The functionality offers the option of including in the export only the data, or the data and the actions.

Rename Use this function to change the name and description of the workflow.

Clone This button creates a clone of the workflow with the a name containing the prefix *Copy of*. Once the operation is executed, the workflow is available in the home screen (link in the upper left corner of the screen).

Flush data table This function deletes the data associated with the workflow. It maintains the set of attributes and the actions, but it removes the conditions and filters from all the actions.

Delete This function deletes completely the workflow from the platform.

Additionally, the following two tabs offer the operations to manage two entities: attributes and sharing.






Workflow attributes You can define a set of *attributes* in the workflow. This is simply a set of pairs *name, value* that you can use to have a single place where a value is defined and then reused in several other locations. For example, the name of the course is probably going to appear in various communications with the learners. If you define the attribute *Course_name* with that value, you can then refer to the attribute and it will be replaced by its value.

Share workflow You may share a workflow with other instructors in the platform. The *Share* button will allow you to add/remove other users to this list. The other users will not be able to flush the data or delete the workflow. Whenever you open a workflow, it becomes unavailable for the other users with whom it is being shared until you either select another workflow or your session expires.

1.5.18 Column operations

Log into the platform and open a workflow that has data in the table. Click in the *Settings* link in the top-bar menu. Select the option *Column operations*. The next page shows the information about the columns:

The buttons at the top of the page are used to perform several operations over the workflow.






Settings

BIOL1011

Column Operations

Next Step: Click in the [Actions](#) menu at the top bar to create actions to execute with the data in the table.

+ Column
Manage table data

Show 10 entries Search:

#	Name	Description	Type	Key?	
1	SID		%	✓	...
2	Identifier		I	✓	...
3	email		I	✓	...
4	Surname		I		...
5	GivenName		I		...
6	MiddleInitial		I		...
7	Full name		I	✓	...
8	Gender		I		...
9	Course Code		I		...
10	Program		I		...

Showing 1 to 10 of 67 entries

Previous
1
2
3
4
5
6
7
Next

You may drag and drop the columns from the left-most cell to change their position in the workflow.

© OnTask Project 2019
Version: B.4.0.1

Adding columns

The *Column* button offers the following options:

Add a new column This function allows you to manually add a new column to the data table. You need to provide the name, a description (optional), the type of data (one of string, number, datetime, or boolean), the position where this column is inserted in the workflow, a date/time window when the columns is visible, an optional comma-separated list of possible value (useful to restrict the values), and an optional initial value.

Add a formula-based column This function is to create a new column containing the result of the data from existing columns combined with certain basic operations such as *maximum*, *minimum*, *sum*, *product*, *mean*, *median*, etc. The way it works is by selecting the operation and a subset of existing columns to use as operands.

Add a column with random values This function creates a new column with values randomly taken from a pre-defined collection.

Uploading or Merging additional data

The button labeled *Manage table data* allows to execute the operation to either upload, merge new data in the table, or execute an existing plugin.

1.5.19 Personalized JSON action








The second type of action available in OnTask is called *personalized JSON*. Create a new action and select this type from the pull-down menu when introducing the name and description.

The screenshot shows a modal dialog titled "Create a new action" with a close button (X) in the top right corner. The dialog contains three input fields: "Name*" with the text "Send JSON to remote server", "Description" with the text "Send a JSON object to a remote server (outside this platform)", and "Action type*" with a dropdown menu showing "Personalized JSON". At the bottom right of the dialog are two buttons: "Cancel" and "Create action" (which has a blue checkmark icon).

The following screen contains the editor for these actions.

The content is divided into three areas (similar to the ones used to edit *personalized text actions*).

1. This section allows to define a *filter*, or a condition to select a subset of the learners for which this action will be considered.
2. This section contains the conditions to be used to conform the personalized JSON object. Two conditions have been defined in the example. The first one with name `Less than 50 in the midterm` is exactly stating that condition, those students for which the score in the midterm has been less than 50. The second condition



BIOL1011

Send JSON to remote server

Personalized JSON Object

Filter Learners (Empty)

Text Conditions (2)

- Insert Column Value -

- Use condition in highlighted text -

```
{
  "sid": {{ SID }},
  "midterm_total": {{ Total }},
  "msg":
    {% if Less than 50 in the midterm %}"Message number 1"{% endif %}
    {% if More or equal to 50 in midterm %}"Message number 2"{% endif %}
}
```

Target URL

http://127.0.0.1|

Preview

Save

Close

© OnTask Project 2019

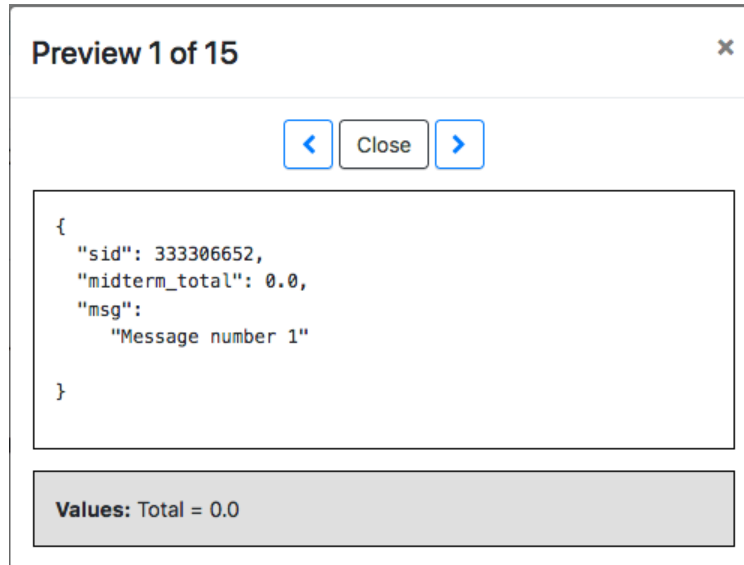
Version: B.4.0.1

is the complementary, those students for which the midterm score has been greater or equal than 50. These conditions are used in the definition of the object in the screen area below.

3. This area is JSON object editor. In the previous figure you see an example of an object that contains three string/value pairs. The first two values are extracted from columns `SID` and `Total` respectively. The last field with name `msg` is defined using the conditions created in the second areaw of this screen.

The *Target URL* field captures the URL to use to send these JSON objects. The action will not be executed unless there is a non-empty value in this field.

As in the case of the *personalized text actions*, the *Preview* button at the bottom of the screen allows you to preview the resulting JSON objects for the selected rows of the data table.








1.5.20 Scheduling Actions

Log in the platform, open a workflow with data and actions, and click in the *Actions* link on the top menu. Sometimes the execution of the actions that produce personalized content (either text or a JSON object) need to be scheduled for some specific point in time and proceed without any instructor intervention. Choose one of these actions and in the *Actions* page click in the link *More* in the operations column. The first operation that appears is *Schedule*. Click on that link.

Scheduling Personalized Text Actions

If the action is a personalized text, the screen requests information to execute the action but including a field stating when to execute it.

In the case of an action that produces personalized text, the *scheduling* needs a name, description (optional), the column that contains the email addresses, the date/time of the execution (in the future), the email subject, comma separated list of CC and BCC email addresses, the choice if a confirmation email is sent, the option to track the emails, and the possibility of checking the list of emails to see if any of them should be excluded. After introducing this information the request to execute the action is submitted to the system.

 Settings

BIOL1011

Schedule Action Execution


Time now is Feb. 19, 2019, 11:57 a.m.

Midterm comments


Name*

Description

When to execute this action*



Column in the table containing the email*



Email subject*

Comma-separated list of CC Emails

Comma-separated list of BCC Emails

☐ Check/exclude email addresses before scheduling?

☐ Send you a confirmation email

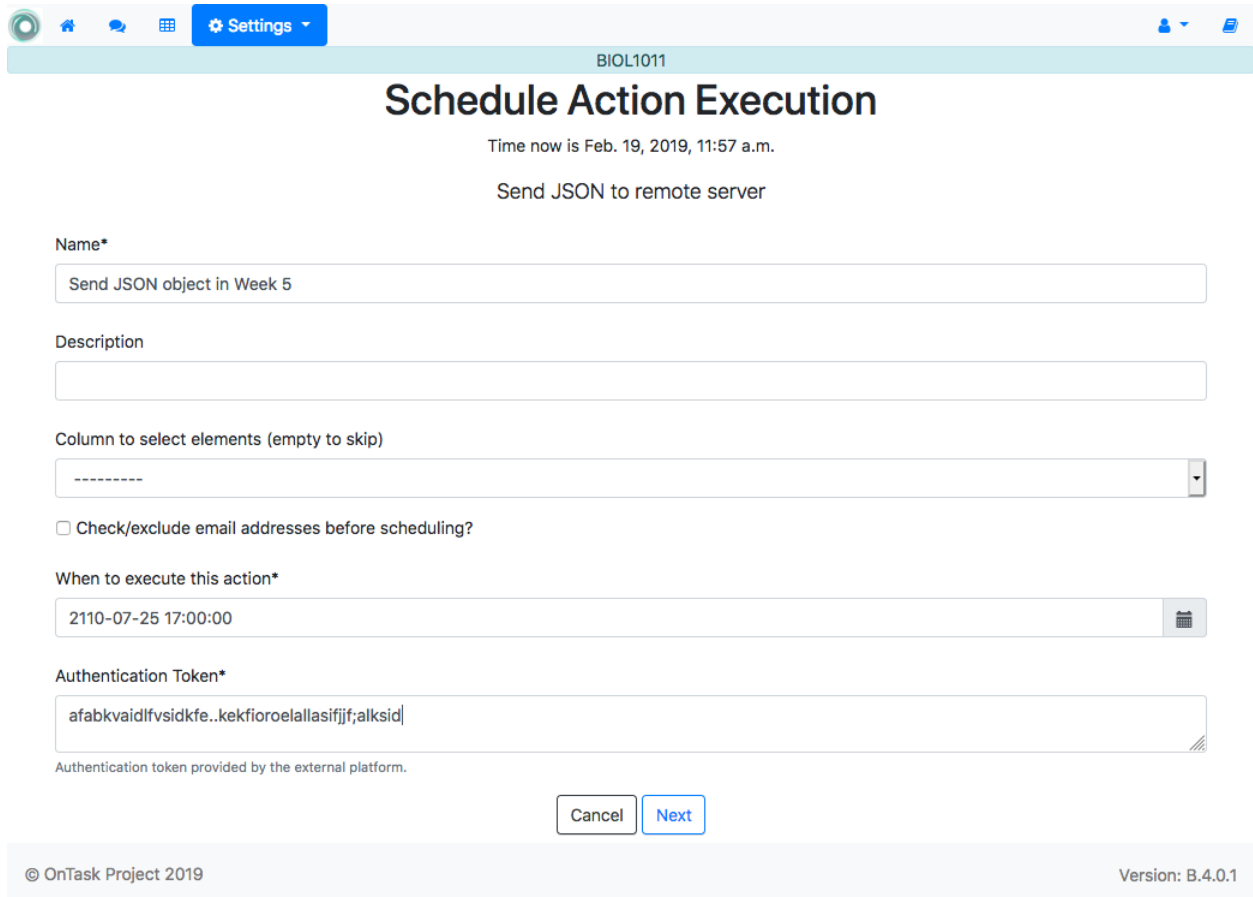
☒ Track if emails are read?

© OnTask Project 2019

Version: B.4.0.1

Scheduling Personalized JSON Actions

If the action is a personalized JSON, the screen requests information to execute the action but including a field stating when to execute it.



The screenshot shows the 'Schedule Action Execution' form in the OnTask web application. The interface includes a top navigation bar with a 'Settings' button and a user profile icon. The main heading is 'Schedule Action Execution' with a subtitle 'Send JSON to remote server'. The form contains several fields: 'Name*' with the value 'Send JSON object in Week 5', 'Description' (empty), 'Column to select elements (empty to skip)' (empty), a checkbox for 'Check/exclude email addresses before scheduling?' (unchecked), 'When to execute this action*' with the value '2110-07-25 17:00:00', and 'Authentication Token*' with a long alphanumeric string. Below the token field is a note: 'Authentication token provided by the external platform.' At the bottom of the form are 'Cancel' and 'Next' buttons. The footer shows '© OnTask Project 2019' and 'Version: B.4.0.1'.

In the case of an action that produces a personalized JSON, the *scheduling* needs a name, description (optional), a column used to select/deselect the elements, and an authentication token to send to the destination platform.

Checking the scheduled execution

All the operations schedule for execution can be verified, edit and deleted from the link *Scheduler* through the top menu.

1.5.21 Merging Data

Work in progress

1.5.22 Download a ZIP (and upload it in Moodle)

Log into the platform, open a workflow that contains data and a personalized action, click in the *Actions* link in the top menu.

BIOL1011

Scheduled Actions*

Show 10 entries Search:

Name ↑↓	Action ↑↓	Scheduled ↑↓	Status ↑↓	
Send Emails after week 3	Midterm comments	07/05/2110 5:30 p.m.	Pending	
Send JSON object in Week 5	Send JSON to remote server	07/25/2110 5 p.m.	Pending	

Showing 1 to 2 of 2 entries Previous 1 Next

*The execution of some of the actions can be scheduled for a time in the future. This page shows those planned executions. To schedule the execution of one action, go to the [Actions](#) menu and select the operation *Schedule* for an action.

© OnTask Project 2019 Version: B.4.0.1

The *personalized text actions* allow you to define a text with parts that are included/excluded depending on the value of the conditions. OnTask then allows you to *send those texts as individual emails*. However, you may want to extract those texts in HTML and process them with a different platform (for example Moodle). You can do this clicking in the *Zip* button shown in the *Operations* column for some of the actions.

Click in this button for any of the personalized text action. The next page includes a form to collect the following information:

1. One of the key columns (unique values in all rows) to use as file name prefix.
2. Second (optional) column to use in the file name.
3. The file name suffix. If it is left empty `feedback.html` will be used.
4. Check the next field **if the ZIP is going to be used to be uploaded as Moodle feedback**.
5. Check the last field if you want to review the users for which the file will be included in the ZIP (see *Final look before pushing the GO button*).

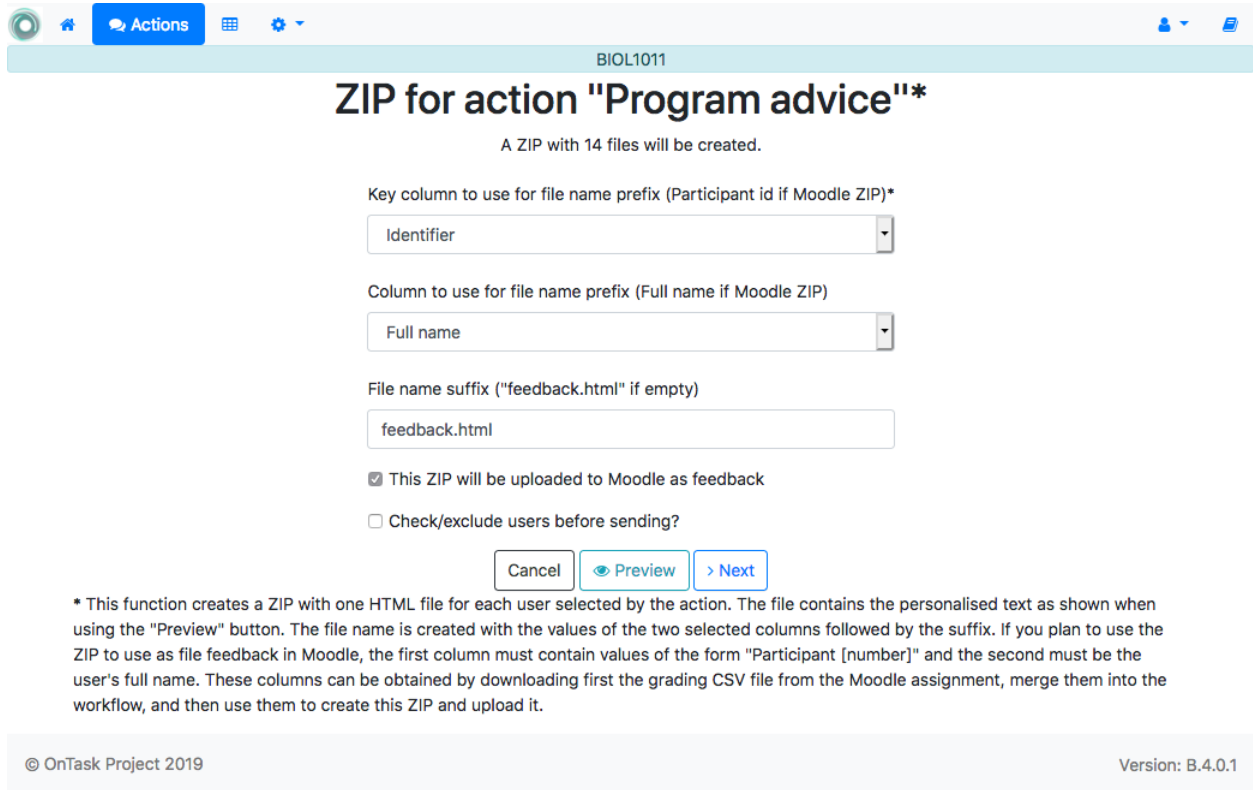
Click in the button *Preview* at the bottom of the page to see the final appearance of the text in the file and the values of the columns used to configure the text. Click in the arrows next to the *Close* button to traverse the files and verify that they are properly composed.

Use the ZIP file to upload feedback in Moodle

If the ZIP created using this functionality has the file names created with certain specific structure, it can be used to upload the personalized messages as part of a Moodle assignment. The workflow data table must have two columns: one with name `Identifier` with values with structure `Participant N` where `N` is a unique number per row, and a column with name `Full name` with the learner's full name. These columns can be obtained *downloading the grading worksheet* from the screen showing all the submissions in a Moodle assignment.

Store the CSV file obtained from this page. We are now going to *merge* the content of the Moodle CSV with the data already existing in the data table. In the *Actions* page, click in the link with name *Manage table data* and then in the *Upload or merge* option. Next, click in the *CSV Upload/Merge* button. Select the CSV grading sheet downloaded from Moodle.

Go to the next step and select only the columns `Identifier`, `Full name` and `Email address`. This last one we will use as the key column to merge with the existing email column. In order not to create an extra column, rename `Email address` to `email` as shown in the next figure:



BIOL1011

ZIP for action "Program advice"*

A ZIP with 14 files will be created.

Key column to use for file name prefix (Participant id if Moodle ZIP)*

Identifier

Column to use for file name prefix (Full name if Moodle ZIP)

Full name

File name suffix ("feedback.html" if empty)

feedback.html

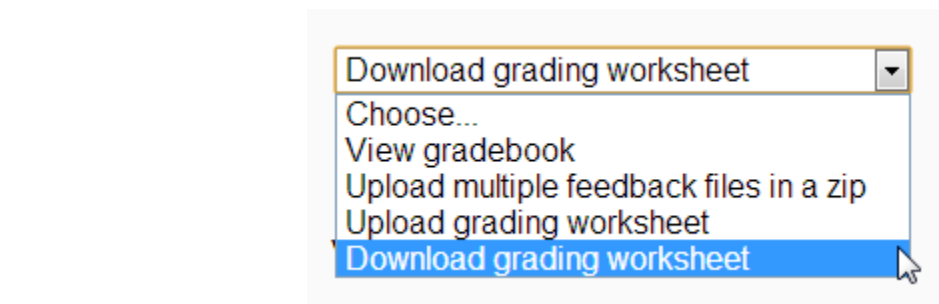
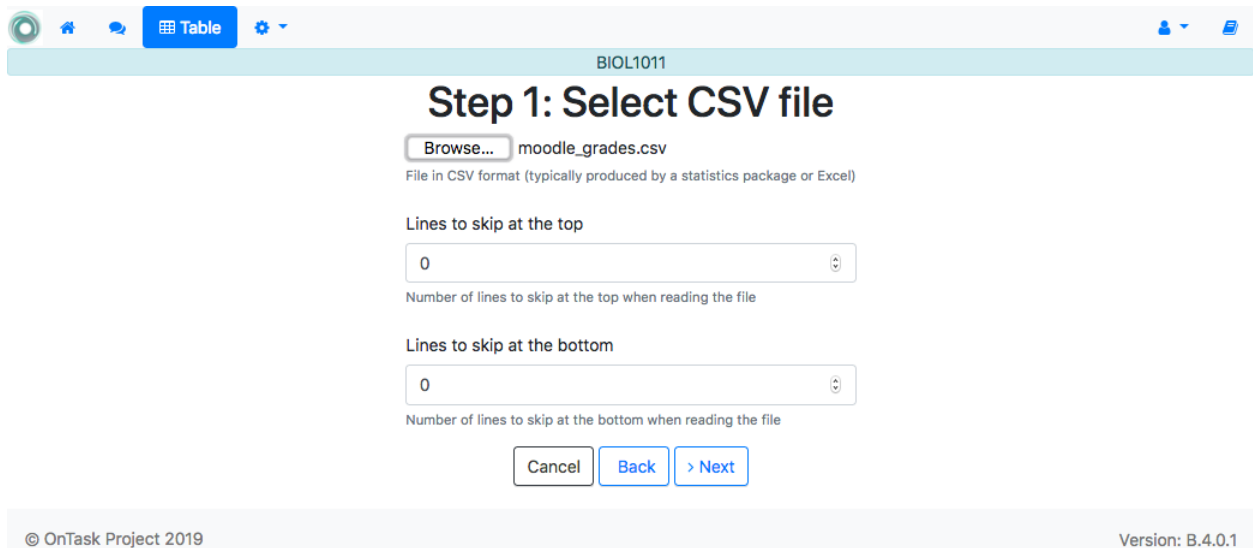
☒ This ZIP will be uploaded to Moodle as feedback

☐ Check/exclude users before sending?

Cancel Preview > Next

* This function creates a ZIP with one HTML file for each user selected by the action. The file contains the personalised text as shown when using the "Preview" button. The file name is created with the values of the two selected columns followed by the suffix. If you plan to use the ZIP to use as file feedback in Moodle, the first column must contain values of the form "Participant [number]" and the second must be the user's full name. These columns can be obtained by downloading first the grading CSV file from the Moodle assignment, merge them into the workflow, and then use them to create this ZIP and upload it.

© OnTask Project 2019 Version: B.4.0.1

BIOL1011

Step 1: Select CSV file

Browse... moodle_grades.csv

File in CSV format (typically produced by a statistics package or Excel)

Lines to skip at the top

0

Number of lines to skip at the top when reading the file




Lines to skip at the bottom

0




Number of lines to skip at the bottom when reading the file

Cancel Back > Next

© OnTask Project 2019 Version: B.4.0.1



Table



BIOL1011

Step 2: Select Columns

Cancel Back > Next

Load	Name	Rename to	Type	Key
<input checked="" type="checkbox"/>	Identifier	<input type="text" value="Identifier"/>	string	Keep? <input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Full name	<input type="text" value="Full name"/>	string	Keep? <input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Email address	<input type="text" value="email"/>	string	Keep? <input checked="" type="checkbox"/>
<input type="checkbox"/>	Status	<input type="text" value="Status"/>	string	
<input type="checkbox"/>	Grade	<input type="text" value="Grade"/>	number	
<input type="checkbox"/>	Maximum Grade	<input type="text" value="Maximum Grade"/>	number	
<input type="checkbox"/>	Grade can be changed	<input type="text" value="Grade can be changed"/>	string	
<input type="checkbox"/>	Last modified (submission)	<input type="text" value="Last modified (submission)"/>	string	
<input type="checkbox"/>	Last modified (grade)	<input type="text" value="Last modified (grade)"/>	datetime	

Cancel Back > Next

© OnTask Project 2019

Version: B.4.0.1

Select `email` as the key for the existing table and `Email address` as the key column for the new table.

Step 3: Select Keys and Merge Option

Cancel Back > Next

Key Column in Existing Table*

email

Key column in the existing table to match with the new table.

Key Column in New Table*

email

Key column in the new table to match with the existing table.

Method to select rows to merge/update*

4) Select only the rows with keys in the new table

Select one method to see detailed information

Selects only the rows with key values present in the new table. This method drops those rows from the existing table with key values not in the new table (rows 1 and 2 in the figure below). The cells in the result rows are updated only if the new value is non-empty (cell 4B remains with value *b*).

Update/Merge

	K	A	B
✗ 1		a	b
✗ 2		a	b
✓ 3		a	b
✓ 4		a	b

Existing

	K	B	C
3		b'	c
4			c
5		b'	c
6			c

New

In New

	K	A	B	C
3		a	b'	c
4		a	b	c
5			b'	c
6				c

Result

Cancel Back > Next

© OnTask Project 2019

Version: B.4.0.1

Verify that you are merging two columns and using `email` as the key.





Click finish and verify that the table in the workflow contains two additional columns: `Identifier` and `Full name`. Go to the *Actions* page and click in the link to generate a ZIP file for one of the personalized text actions. Select now the column `Identifier` as the first name prefix and the column `Full name` as the second column prefix. Choose additional file name suffix (or leave it empty), and *click the option for the ZIP to be uploaded to Moodle*.

Once created, go to the page showing the submissions for an assignment in Moodle and choose the option to *Upload multiple feedback files in a zip*.



Upload the zip just created and each file will be placed as the feedback file for each student (independently if there is a submission or not).

1.5.23 Uploading Data from a Remote Database

Work in progress

**Table**

BIOL1011







Step 4: Review and confirm

Cancel

Back

> Finish

Existing Table		New Data
email (Key)		email (Key)
Attendance		
Contributions		
Contributions 2		
Contributions 3		
Contributions 4		
Contributions 5		
Correct_1_W2		
Correct_1_W3		
Correct_1_W4		
Correct_1_W5		
Correct_2_W2		
Correct_2_W3		
Correct_2_W4		
Correct_2_W5		
Course Code		
Days online		
Days online 2		

  **Actions**  

BIOL1011

ZIP for action "Program advice"*

A ZIP with 14 files will be created.

Key column to use for file name prefix (Participant id if Moodle ZIP)*

Identifier

Column to use for file name prefix (Full name if Moodle ZIP)

Full name


File name suffix ("feedback.html" if empty)

feedback.html

☒ This ZIP will be uploaded to Moodle as feedback

☐ Check/exclude users before sending?

Cancel

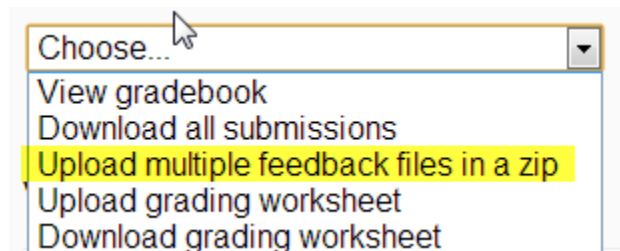
 Preview

> Next

* This function creates a ZIP with one HTML file for each user selected by the action. The file contains the personalised text as shown when using the "Preview" button. The file name is created with the values of the two selected columns followed by the suffix. If you plan to use the ZIP to use as file feedback in Moodle, the first column must contain values of the form "Participant [number]" and the second must be the user's full name. These columns can be obtained by downloading first the grading CSV file from the Moodle assignment, merge them into the workflow, and then use them to create this ZIP and upload it.

© OnTask Project 2019

Version: B.4.0.1



1.5.24 Plugins: Write your own data processing code

Example: A Predictive Model

Suppose that your favorite data analyst has processed the data set and created a predictive model that estimates the score of the final exam based on the value of the column *Contributions* applying the following linear equation:

```
final exam score = 3.73 * Contributions + 25.4
```

You would like to incorporate this model to the workflow and use the predicted final exam score as another column to create conditions and personalize content. One way to achieve this is by creating a plugin that given the two coefficients of a linear model (in the example 3.73 and 25.4) returns a new data set with a column with the values obtained using the corresponding equation. In order for the plugin to comply with the *requirements*, one possible definition would be:

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals, print_function

import pandas as pd

# The field class_name contains the name of the class to load to execute the
# plugin.
class_name = 'OnTaskTestPlugin'

class OnTaskTestPlugin:
    """
    Example of a class that implements the OnTask plugin interface. The
    objects of this class have to provide the following elements:

    1. name: Plugin name show to the users.

    2. description_txt: A string with the detailed description of what the
    plugin does

    3. input_column_names: A potentially empty list of column names (strings).
    If the list is empty, the columns are selected by the user at execution
    time.

    4. output_column_names: Non empty list of names (strings) of the columns
    to be used for the output of the transformation.

    5. parameters: an optionally empty list with tuples with the following
    structure:

    ('name', type, [list of allowed values], initial value, help_text)

    These elements will be requested from the user before executing the
    plugin through a form. The conditions on these values are:

    - name must be a string
    - type must be a string equal to "integer", "double", "string",
      "datetime" or "boolean".
    - The list of values is to restrict the
      possible values
    - The initial value must be of the type specified by the second
      element.
    - Help_text a string to show as help text
```

(continues on next page)

(continued from previous page)

```

6. method "run" that receives:
- a pandas data frame with the data to process
- a string with the name of the key column that will be used to merge
  the result.
- A dictionary of pairs (name, value) with the parameters described in
  the previous element.

and returns a result Pandas data frame. This frame must have one
column with the key column name provided so that it can be properly
merged with the existing data.
"""

def __init__(self):
    self.name = 'Test Plugin 1 Name'
    self.description_txt = 'Test Plugin 1 Description Text'
    self.input_column_names = list()
    self.output_column_names = ['RESULT 1', 'RESULT 2']
    self.parameters = [
        ('param string', 'string', ['v1', 'v2'], 'v1', 'help param string'),
        ('param integer', 'integer', [], None, 'help param integer'),
        ('param double',
         'double',
         [1.2, 2.2, 3.2],
         None,
         'help param double'),
        ('param boolean', 'boolean', [], True, 'help param boolean'),
        ('param datetime',
         'datetime',
         [],
         '2018-05-25 18:03:00+09:30',
         'help param datetime'),
        ('param datetime2',
         'datetime',
         [],
         '2018-05-25 18:03:00+09:30',
         'help param datetime'),
    ]

def run(self, data_frame, merge_key, parameters=dict):
    """
    Method to overwrite. Receives a data frame with a number of columns
    stipulated by the num_column_input pair, the name of a key column and a
    dictionary with parameters of the form name, value.

    Runs the algorithm and returns a pandas data frame structure that is
    merged with the existing data frame in the workflow using the merge_key.

    :param data_frame: Input data for the plugin
    :param merge_key: Name of the column key that will be used for merging
    :param parameters: Dictionary with (name, value) pairs.

    :return: a Pandas data_frame to merge with the existing one (must
    contain a column with name merge_key)
    """

    # Extract the key column from the given data frame

```

(continues on next page)

(continued from previous page)

```
result = pd.DataFrame(data_frame[merge_key])

# Process the given data and create the result
result[self.output_column_names[0]] = 1
result[self.output_column_names[1]] = 2

return result
```

1.6 Advanced Features

OnTask is built in [Python](#) using the web framework [Django](#) in combination with some additional libraries such as [Django REST Framework](#), [Pandas](#), etc. The application is available as [open source](#) with [MIT License](#). This means that the advanced users can get a copy of the source code and modify it to suite their needs using the already existing models and functions.

1.6.1 The API (Application Programming Interface)

OnTask is a platform that facilitates the connection between data and the provision of personalised learner support actions. The higher the quality of the data the higher number of possible effective support actions. This means that OnTask should facilitate the connection with already existing data sources so that it can combine data sets and create a comprehensive view of how a learning experience is evolving.

The API is documented online through the URL suffix `apidoc`. The page contains the description of every entry point available with the required parameters.

When manipulating the elements in the table there are two versions of the basic operations (create a table, update a table, merge).

1. **Pandas Version.** This version handles the encoding of data frames using the pandas pickle encoding. This encoding has the advantage that maintains the elements of the dataframe intact. In other words, when the data frame is decoded from the pandas pickle format back to a regular dataframe, the same initial dataframe is obtained.
2. **JSON Version.** This version encodes the dataframes in JSON. The problem with this format is that values such as NaN and NaT (not a time) are not allowed by JSON, and they are substituted by empty strings. This change may have significant effects on the dataframe, specially on the types of the columns. If there is a dataframe with a column of type datetime and with any element with value NaT and it is first extracted through the JSON interface, and then uploaded again, the NaT value is transformed into an empty string, and Pandas will no longer recognise that column as datetime, but instead it will render the column of type string. This may have also an effect on how rules and actions are evaluated.

If OnTask is hosted in a server like Apache, you may need to check your configuration so that it allows the processing of the `HTTP_AUTHORIZATION` headers. For example, the module WSGI for Apache by default filters out this header.

[Changelog](#)