# Portfolio Documentation

***Release 2018-10-18***

**Ida Bergquist Eric Jönsson**

**Oct 18, 2018**

# CONTENTS:

1 data 1

2 main 3

3 forms 7

Python Module Index 9

Index 11

CHAPTER

# ONE

## DATA

data.**get_project**(*db*, *i_d*)
Please refer to LiU Documentation

data.**get_project_count**(*db*)
Please refer to LiU Documentation

data.**get_searchfields**(*db*)
Get all the search fields

Function returns all the searchable fields present in specified database db.

**Parameters** **db** (*list*) – A list object representing the database.

**Returns** **search_fields** – A list of all the searchable fields.

**Return type** list

data.**get_technique_stats**(*db*)
Please refer to LiU Documentation

data.**get_techniques**(*db*)
Please refer to LiU Documentation

data.**load**(*filename*)
Please refer to LiU Documentation

data.**load_users**(*filename*)
Loads users from file

Loads JSON formatted user data from a specified file.

**Parameters** **filename** (*str*) – The file to read from.

**Returns** **db** – A dictionary of all the users.

**Return type** dict

data.**remove_project**(*db*, *i_d*)
Removes a project

Function removes a project with a certain ID from the database list.

**Parameters**

- **db** (*list*) – The list to remove the project from.
- **i_d** (*int*) – The project_id to remove, as an int.

data.**save**(*database*, *filename*)
Saves the database to file

Function takes in a database in the form of a list and saves it to the desired filename in the current working directory.

**Parameters**

- **database** (*list*) – The database to be modified, as a list.
- **filename** (*str*) – The file to save to, as as string.

data.**search**(*db*, *sort_by='start_date'*, *sort_order='desc'*, *techniques=None*, *search=None*, *search_fields=None*)

Please refer to LiU Documentation

CHAPTER

# TWO

## MAIN

**class** main.**User**(*user_id*)

Flask login user

This is a basic user template for flask_login that inherits from flask_login's UserMixin class.

**id**

An integer representing the user's id.

**Type** int

main.**edit**()

Edit view

Returns a view of the edit page. The edit page is a list of all the currently existing projects in the database.

**Returns**

**Return type** render_template

main.**get**()

This is an easter egg.

It has goats.

**Returns** Some HTML.

**Return type** str

main.**get_images**(*i_d*)

Get all images from a project

This function returns every image present in the project id's corresponding image folder on the server.

**Parameters** **i_d** (*int*) – The project ID.

**Returns** **images** – Every image's relative path.

**Return type** list

main.**imggetter**()

Context processor

This function lets Flask access any functions specified within its return dictionary from any context, so that they can be called from within Jinja templates.

**Returns** **dict** – Dictionary of all functions and variables to be accessible in every context.

**Return type** dict

main.**index**()
Index view

Returns a view for the index page. Fetches all the projects in the database and passes them on to render_template function together with the index.html template page.

**Returns**

**Return type** render_template

main.**invalid_login**(*error*)
401 view

Returns the 401 view for not authenticated.

**Returns**

**Return type** render_template

main.**list**()
List view

Returns a view of the list page. Utilises the HTML POST method to request data from user for use in database search.

If the method is POST the function will request projects from the data.search function and pass them on to the render_template method. If the method is GET it will simply return all projects from data.search to render_template.

**Returns**

**Return type** render_template

main.**load_user**(*user_id*)
Callback for login_user

This is a callback function for flask login. It's called every time we try to log in a user with flask_login.login_user.

**Parameters** **user_id** (*int*) – User representation.

main.**login**()
Login view

Returns a view of the login page. The login page lets you log in if you aren't yet authenticated. If the login is successful, it redirects to the edit view. HTML GET and POST methods are used to get the data from the end user.

**Returns**

**Return type** render_template

main.**logout**()
Logout view

Returns logout view as well as logs the user out with flask_login. logout_user().

**Returns**

**Return type** string

main.**modify**(*project_id*)
Modify view

Returns a view of the modify page. The modify page supports both HTML GET and POST methods. The GET method is used to request the desired project, and is passed into the modify function through Flask's routing

method. This data is then used to populate the fields on the page. The POST method is used to populate the WTForm instance, which is then used to edit the actual database.

**Parameters** **project_id** (*int*) – The integer representing the desired project_id key in the database.

**Returns**

**Return type** render_template

main.**page_not_found**(*error*)
404 view

Returns 404 view for page not found.

**Returns**

**Return type** render_template

main.**project**(*project_id*)
Project view

Returns a view of the project page. Takes as input a GET variable from Flask's route method and requests the desired project from the database based upon this integer value.

**Parameters** **project_id** (*int*) – The integer representing the desired project_id key in the database.

**Returns**

**Return type** render_template

main.**techniques**()
Technique view

Returns a view of the technique page with each technique on its own row, together with its relevant projects.

**Returns**

**Return type** render_template

CHAPTER

# THREE

# FORMS

**class** forms.**ModifyForm**(**args*, ***kwargs*)

Bases: wtforms.form.Form

Form for modifying a project.

It has several fields of various types (StringField, IntegerField, FileField, etc.) that correspond to the indexes in the project's dictionary, which are specified in the project specification. Please refer to: LiU Documentation

**Parameters**

- ***args** (*args) – Any number of positional arguments.
- ****kwargs** (**kwargs) – Any number of keyword arguments.

**class** forms.**ModifyFormAdd**(**args*, ***kwargs*)

Bases: *forms.ModifyForm*

Form for adding new projects

Like ModifyForm, except it inherts a validations method for the project_id field in order to avoid conflicts with other projects already in the database.

**Parameters**

- ***args** (*args) – Any number of positional arguments.
- ****kwargs** (**kwargs) – Any number of keyword arguments.

# PYTHON MODULE INDEX


d

data, 1

f

forms, 7

m

main, 3

# INDEX


## D

data (module), 1

## E

edit() (in module main), 3

## F

forms (module), 7

## G

get() (in module main), 3
get_images() (in module main), 3
get_project() (in module data), 1
get_project_count() (in module data), 1
get_searchfields() (in module data), 1
get_technique_stats() (in module data), 1
get_techniques() (in module data), 1

## I

id (main.User attribute), 3
imggetter() (in module main), 3
index() (in module main), 3
invalid_login() (in module main), 4

## L

list() (in module main), 4
load() (in module data), 1
load_user() (in module main), 4
load_users() (in module data), 1
login() (in module main), 4
logout() (in module main), 4

## M

main (module), 3
modify() (in module main), 4
ModifyForm (class in forms), 7
ModifyFormAdd (class in forms), 7

## P

page_not_found() (in module main), 5
project() (in module main), 5

## R

remove_project() (in module data), 1

## S

save() (in module data), 1
search() (in module data), 2

## T

techniques() (in module main), 5

## U

User (class in main), 3