
Flashmap server Documentation

Release 1.0

M.C. van den Enk

Mar 30, 2017

CONTENTS

1	concept_map module	1
2	consumer module	3
3	edge module	5
4	flashcard module	7
5	flashcard_instance module	9
6	flashmap_instance module	11
7	handler module	13
8	instance module	15
9	logentry module	17
10	node module	19
11	questionnaire module	21
12	questionnaire_item module	23
13	questionnaire_response module	25
14	response module	27
15	session module	29
16	test module	31
17	test_flashcard_response module	33
18	test_item module	35
19	test_item_response module	37
20	user module	39
21	Indices and tables	43
	Python Module Index	45

CONCEPT_MAP MODULE

```
class concept_map.ConceptMap (*args, **values)
    Bases: mongoengine.document.Document
```

A class representing a concept map

Variables

- **nodes** – a list of nodes (by default all existing node documents)
- **edges** – a list of edges (by default all existing edge documents)

```
find_nodes (edges)
```

Returns the from and to nodes given a list of edges

Parameters **edges** (*list* (*Edge*)) – The list of edges for which to find the nodes

Returns The list of nodes referred to in the edges

Return type *list(Node)*

```
find_prerequisites (postreq, prereqs, sources)
```

Return a list of parent edges given a certain edge from a list of edges, filtered by a list of sources

Parameters

- **postreq** (*Edge*) – The edge which is currently investigated for parent edges
- **prereqs** (*list* (*Edge*)) – A list of already found parent edges (starts usually empty, necessary for recursion)
- **sources** (*list* (*string*)) – A list of the currently read sources, edges which have a source not included in this list will not be included in the resulting list

Returns A list of edges which are prerequisites from edge

Return type *list(edge)*

```
find_siblings (edge, sources, partial_edges)
```

Return a list of edges which are siblings of the given edge

Parameters

- **edge** (*Edge*) – The edge investigated for siblings
- **sources** (*list* (*string*)) – The sources to filter on when looking for siblings
- **partial_edges** (*list* (*Edge*)) – A list of edges to filter on when looking for siblings

Returns A list of edges which are siblings of edge

Return type *list(edge)*

get_partial_map (*edge, sources*)

Returns a concept map containing only the parent and sibling edges together with the referred nodes

Parameters **edge** (*Edge*) – The input edge

Returns A concept map containing parent and sibling edges of edge together with the referred nodes

Return type *ConceptMap*

to_dict ()

Returns a dictionary representation of this object

The representation is compatible for use with vis.js, with ‘nodes’ entries containing an ‘id’ and ‘label’, and ‘edges’ entries containing an ‘id’, ‘label’, ‘from’, ‘to’, and an additional ‘source’ entry

Result The dictionary representation

Return type *dict*

CONSUMER MODULE

class `consumer.Consumer`

Bases: `object`

This is the class from which the program is controlled. It can be used together with the *handler* module in order to communicate with an external client over a websocket

Variables

- *concept_map* – The concept map object containing references to nodes and edges
- **SOURCES** – All of the sources referenced to in the edges of the concept map
- *user* – The active user

add_source (*source*)

Adds a read source to the active user

Parameters **source** (*string*) – The source to be added

authenticate (*name*)

A function to either set self.user to an existing *user.User* or to a new User based on the given name

Parameters **name** (*str*) – The username

check_prerequisites ()

Checks whether the user still has to fill in forms and returns the appropriate message

Returns A dict containing the appropriate keyword and data for this user

Return type `dict`

consumer (*keyword, data*)

Pass data to the function corresponding to the provided keyword for the provided user

Parameters

- **keyword** (*str*) – the keyword for which function to use
- **data** (*dict(str, str or dict)*) – the data necessary for executing the function

Returns Contains the keyword and data to send over a websocket to a client

Return type `dict(str, str or dict)`

create_questionnaire ()

Creates a questionnaire for this user (using `user.create_questionnaire()`)

Returns A dict object fit for sending to the user

Return type `dict`

create_test()

Creates a test for this user (using `user.create_test()`)

Returns A dict object fit for sending to the user

Return type `dict`

provide_learned_items()

Provides an overview of all learning

Returns A partial concept map containing all instances for this user or a message containing progress information

Return type `dict`

provide_learning()

Provides a dict containing relevant information for learning

Provides a dict containing the keyword “NO_MORE_INSTANCES”, “READ_SOURCE-REQUEST”, or “LEARNING-RESPONSE” and relevant data (the source string for “READ_SOURCE-REQUEST” or either the output of `ConceptMap.to_dict()` with an added ‘learning’ entry or the output of `Flashcard.to_dict()` for “LEARNING-RESPONSE” with an added condition entry)

Returns A dict containing ‘keyword’ and the relevant ‘data’ described above

Return type `dict`

validate(responses)

Adds responses to certain instances

Parameters **responses** (`list(dict)`) – A list of responses containing an instance id and a boolean correctness value

EDGE MODULE

```
class edge.Edge (*args, **values)  
    Bases: mongoengine.document.Document
```

A class representing an edge from a concept map

Variables

- **from_node** – The parent node of the edge
- **to_node** – The child node of the edge
- **label** – A label describing the relation between from_node and to_node
- **source** – The source where this edge is described (e.g. paragraph 13.2 from Laagland)

FLASHCARD MODULE

```
class flashcard.Flashcard(*args, **values)
```

```
    Bases: mongoengine.document.Document
```

A class representing a flashcard :cvar question: The question on the front side of the flashcard :type question: StringField :cvar answer: The answer on the back side of the flashcard :type answer: StringField :cvar sources: The sources where this flashcard are described (e.g. paragraph 13.2 of Laagland) :type sources: ListField(StringField) :cvar response_model: A list consisting of parts of valid responses to the question (for the test matrix) :type response_model: ListField(StringField)

```
to_dict()
```

Returns a dictionary representation of this object

It contains an 'id', 'question', 'answer', and 'sources' entry

Returns The dictionary representation of this object

Return type dict

FLASHCARD_INSTANCE MODULE

```
class flashcard_instance.FlashcardInstance(*args, **kwargs)
```

Bases: *instance.Instance*

A class for storing responses from the flashmap system

Variables **reference** – The flashcard to which this instance refers

FLASHMAP_INSTANCE MODULE

class flashmap_instance.**FlashmapInstance** (*args, **kwargs)

Bases: *instance.Instance*

A class for storing responses from the flashmap system

Variables **reference** – The edge from the concept map to which this instance refers to

HANDLER MODULE

`handler.handler (websocket, path)`

Initiate an asyncio thread which receives messages from a client, parse the json file to an object, pass them to `consumer()` and send the result back to the client

Variables

- **websocket** – the websocket being used for receiving and sending messages to a client
- **path** – the IP address used to host the websocket

INSTANCE MODULE

class `instance.Instance` (**args, **kwargs*)

Bases: `mongoengine.document.EmbeddedDocument`

A class describing a general flash instance, which can either be a `FlashmapInstance` or a `FlashcardInstance`

Variables

- **responses** – A list of responses provided to this instance (an empty list by default)
- **reference** – A reference to either an edge in a concept map or a flashcard (defined within the subclass)
- **due_date** – The date this instance is due for repetition

check_due ()

Checks whether this instance is due for repetition

Returns Whether the due datetime is earlier than the current datetime

Return type `boolean`

finalise_response (*correct*)

Sets the correctness value for the final response and sets the end date to now

Parameters **correct** (*boolean*) – Whether the response was correct

get_exponent ()

Determines the exponent for the rescheduling of this instance

Returns The amount of times this instance was answered correctly since the previous incorrect answer

Return type `int`

schedule ()

Reschedules this instance for review based on the previous responses

start_response ()

Adds a new response to this instance

LOGENTRY MODULE

```
class logentry.LogEntry (*args, **values)  
    Bases: mongoengine.document.Document
```

An object representing a incoming or outgoing network message

Variables

- **user** – The user which was involved with this network message
- **keyword** – The network keyword
- **data** – The dictionary containing the necessary data
- **timestamp** – The time that this message was received or transmitted

NODE MODULE

```
class node.Node (*args, **values)
    Bases: mongoengine.document.Document
    A class for representing nodes in the concept map
    Variables label1 – The label appearing within the node
```


QUESTIONNAIRE MODULE

```
class questionnaire.Questionnaire(pu_items, peou_items, **data)
```

```
    Bases: mongoengine.document.EmbeddedDocument
```

A class representing a stored questionnaire for a user

Variables

- **perceived_usefulness_items** – Responses to the perceived usefulness items from TAM
- **perceived_ease_of_use_items** – Responses to the perceived ease of use item from TAM
- **good** – A description of what was good about the software according to the user
- **can_be_improved** – A description of what could be improved according to the user

```
append_answer(item, phrasing, answer)
```

Appends an answer to an item within the questionnaire

Parameters

- **item** (`QuestionnaireItem`) – The item to which the answer refers
- **phrasing** (`boolean`) – Whether the item is positively (True) phrased or negatively (False)
- **answer** (`string`) – The answer to be appended

QUESTIONNAIRE_ITEM MODULE

```
class questionnaire_item.QuestionnaireItem(*args, **values)  
    Bases: mongoengine.document.Document
```

A class representing a single item on the questionnaire

Variables

- **usefulness** – Defines whether the item is part of the perceived usefulness items (True) or of the perceived ease of use items (False)
- **positive_phrasing** – The version of this item which is positively phrased
- **negative_phrasing** – The version of this item which is negatively phrased

QUESTIONNAIRE_RESPONSE MODULE

```
class questionnaire_response.QuestionnaireResponse(*args, **kwargs)
```

```
    Bases: mongoengine.document.EmbeddedDocument
```

A class for storing singular responses to questionnaire items

Variables

- *questionnaire_item* – The questionnaire item to which this answer refers
- **answer** – The value of the likert-scale rating the user gave to this item (ranges from -2 to 2)
- **phrasing** – Whether this answer refers to the positively (True) or the negatively (False) phrased version of the questionnaire_item

RESPONSE MODULE

```
class response.Response (*args, **kwargs)  
    Bases: mongoengine.document.EmbeddedDocument
```

A class representing a singular response to an `Instance`.

Variables

- **start** – The moment the parent `Instance` was sent to the client
- **end** – The moment the answer from the client was received
- **correct** – Whether the answer to the `Instance` was correct (`True`) or incorrect (`False`)

SESSION MODULE

```
class session.Session(*args, **kwargs)
    Bases: mongoengine.document.EmbeddedDocument
```

A class representing a session the user was logged in

Variables

- **start** – The time that the user logged in
- **end** – The time that the user logged out
- **source_prompted** – Whether the user was asked to have read a certain source from SOURCES
- **browser** – The type of browser used to log in

```
end_session()
    Closes this session
```


TEST MODULE

class `test.Test` (*flashcards*, *items*, *prev_flashcards*=[], *prev_items*=[], ****data**)

Bases: `mongoengine.document.EmbeddedDocument`

A class representing a pre- or posttest the user filled in

Variables

- **test_flashcard_responses** – A list of responses to the flashcard questions on the test
- **test_item_responses** – A list of responses to the item questions on the test

generate_test (*items*, *prev_items*)

A method for taking five random items in a random order from the provided list of items without the items in the previous items

Parameters

- **items** (*list*(`Flashcard`) or *list*(`TestItem`)) – The complete list of items
- **prev_items** (*list*(`Flashcard`) or *list*(`TestItem`)) – The list of items to be excluded from the result

Result A sample of five items from items not included in *prev_items*

Return type *list*(`FlashcardResponse`) or *list*(`TestItemResponse`)

TEST_FLASHCARD_RESPONSE MODULE

```
class test_flashcard_response.TestFlashcardResponse(*args, **kwargs)
    Bases: mongoengine.document.EmbeddedDocument
```

An answer for a flashcard item within a pre- or posttest

Variables

- **answer** – The answer provided by the user
- *flashcard* – The flashcard to which this response refers to

TEST_ITEM MODULE

```
class test_item.TestItem(*args, **values)
    Bases: mongoengine.document.Document
    A class representing an item from a pre- or posttest
```

Variables

- **question** – The question for this item
- **sources** – A list of sources relevant to this question
- **response_model** – A list of the parts of a valid answer used for the test matrix

TEST_ITEM_RESPONSE MODULE

```
class test_item_response.TestItemResponse(*args, **values)
```

```
    Bases: mongoengine.document.Document
```

A class representing singular answers to test items

Variables

- **answer** – The answer to item provided by the user
- **item** – The specific item this response refers to

USER MODULE

```
class user.User(*args, **values)
    Bases: mongoengine.document.Document
```

A class representing a user

Variables

- **name** – The username
- **type** – StringField
- **condition** – The condition of the user (“FLASHMAP” or “FLASHCARD”)
- **type** – StringField
- **birthdate** – The birthdate of the user
- **read_sources** – A list of read sources by the user
- **gender** – The gender of the user (can be either ‘male’, ‘female’, or ‘other’)
- **code** – The code from the user’s informed consent form
- **tests** – The pre- and posttest
- **questionnaire** – The questionnaire
- **instances** – A list of instances storing the flashmap/flashcard data for the user
- **sessions** – A list of past sessions for this user
- **email** – The email address for this user

```
add_new_instance(references)
    Adds a new Instance to this user
```

Parameters **reference** (*list* (Flashcard or Edge)) – A set of flashcards or edges for which to add a new instance

Returns The reference for which a new instance was added

Return type Flashcard or Edge

```
append_questionnaire(responses, good, can_be_improved)
    A method for appending a questionnaire to the user given responses
```

Parameters

- **responses** (*dict*) – A list of dict objects containing a QuestionnaireItem (key = ‘item’), the phrasing (key = ‘phrasing’) and an answer (key = ‘answer’)

- **good** (*string*) – A description of what was good about the software according to the user
- **can_be_improved** (*string*) – A description of what can be improved about the software according to the user

append_test (*flashcard_responses*, *item_responses*)

A method for appending a test to the user given flashcard and item responses

Parameters

- **flashcard_responses** (*dict*) – A list of dict objects containing a Flashcard (key = 'card') and an answer (key = 'answer')
- **item_responses** (*dict*) – A list of dict objects containing a TestItem (key = 'item') and an answer (key = 'answer')

create_questionnaire (*pu_items*, *peou_items*)

A method for creating a new questionnaire

Parameters

- **pu_items** – A list of questionnaire items
- **peou_items** – A list of questionnaire items

Returns A randomised list of questionnaire items

Return type *list(QuestionnaireItem)*

create_test (*flashcards*, *items*)

A method for creating a new test with unique questions

Parameters

- **flashcards** (*list(Flashcard)*) – A list of flashcards from the database
- **items** (*list(TestItem)*) – A list of items from the database

Returns A dict containing a list of FlashcardResponses and TestItemResponses

Return type *dict(string, Response)*

get_due_instance ()

Returns the instance with the oldest due date

Returns Either the instance with the lowest due date or a None object

Return type *Instance*

provide_learned_items ()

To be implemented at the specific subclass

set_descriptives (*birthdate*, *gender*, *code*)

A method for setting the descriptives of the user

Parameters

- **birthdate** (*DateTime*) – The provided birthdate of the user
- **gender** (*string*) – The gender of the user (can be either 'male', 'female', or 'other')
- **code** (*string*) – The code from the informed consent form

start_response (*instance*)

Starts a new response within this instance

Parameters **instance** (*Instance*) – The instance to which the response refers

validate (*instance_id*, *correct*)

Finalises a Response within an existing Instance

Parameters

- **instance_id** – The id of the instance which the response refers to
- **correct** (*boolean*) – Whether the response provided by the user was correct or not

INDICES AND TABLES

- `genindex * modindex * search`

PYTHON MODULE INDEX

C

concept_map, 1
consumer, 3

e

edge, 5

f

flashcard, 7
flashcard_instance, 9
flashmap_instance, 11

h

handler, 13

i

instance, 15

l

logentry, 17

n

node, 19

q

questionnaire, 21
questionnaire_item, 23
questionnaire_response, 25

r

response, 27

s

session, 29

t

test, 31
test_flashcard_response, 33
test_item, 35
test_item_response, 37

u

user, 39

INDEX

A

`add_new_instance()` (user.User method), 39
`add_source()` (consumer.Consumer method), 3
`append_answer()` (questionnaire.Questionnaire method), 21
`append_questionnaire()` (user.User method), 39
`append_test()` (user.User method), 40
`authenticate()` (consumer.Consumer method), 3

C

`check_due()` (instance.Instance method), 15
`check_prerequisites()` (consumer.Consumer method), 3
`concept_map` (module), 1
`ConceptMap` (class in `concept_map`), 1
`Consumer` (class in `consumer`), 3
`consumer` (module), 3
`consumer()` (consumer.Consumer method), 3
`create_questionnaire()` (consumer.Consumer method), 3
`create_questionnaire()` (user.User method), 40
`create_test()` (consumer.Consumer method), 3
`create_test()` (user.User method), 40

E

`Edge` (class in `edge`), 5
`edge` (module), 5
`end_session()` (session.Session method), 29

F

`finalise_response()` (instance.Instance method), 15
`find_nodes()` (concept_map.ConceptMap method), 1
`find_prerequisites()` (concept_map.ConceptMap method), 1
`find_siblings()` (concept_map.ConceptMap method), 1
`Flashcard` (class in `flashcard`), 7
`flashcard` (module), 7
`flashcard_instance` (module), 9
`FlashcardInstance` (class in `flashcard_instance`), 9
`flashmap_instance` (module), 11
`FlashmapInstance` (class in `flashmap_instance`), 11

G

`generate_test()` (test.Test method), 31

`get_due_instance()` (user.User method), 40
`get_exponent()` (instance.Instance method), 15
`get_partial_map()` (concept_map.ConceptMap method), 1

H

`handler` (module), 13
`handler()` (in module `handler`), 13

I

`Instance` (class in `instance`), 15
`instance` (module), 15

L

`LogEntry` (class in `logentry`), 17
`logentry` (module), 17

N

`Node` (class in `node`), 19
`node` (module), 19

P

`provide_learned_items()` (consumer.Consumer method), 4
`provide_learned_items()` (user.User method), 40
`provide_learning()` (consumer.Consumer method), 4

Q

`Questionnaire` (class in `questionnaire`), 21
`questionnaire` (module), 21
`questionnaire_item` (module), 23
`questionnaire_response` (module), 25
`QuestionnaireItem` (class in `questionnaire_item`), 23
`QuestionnaireResponse` (class in `questionnaire_response`), 25

R

`Response` (class in `response`), 27
`response` (module), 27

S

`schedule()` (instance.Instance method), 15

Session (class in session), [29](#)
session (module), [29](#)
set_descriptives() (user.User method), [40](#)
start_response() (instance.Instance method), [15](#)
start_response() (user.User method), [40](#)

T

Test (class in test), [31](#)
test (module), [31](#)
test_flashcard_response (module), [33](#)
test_item (module), [35](#)
test_item_response (module), [37](#)
TestFlashcardResponse (class in test_flashcard_response), [33](#)
TestItem (class in test_item), [35](#)
TestItemResponse (class in test_item_response), [37](#)
to_dict() (concept_map.ConceptMap method), [2](#)
to_dict() (flashcard.Flashcard method), [7](#)

U

User (class in user), [39](#)
user (module), [39](#)

V

validate() (consumer.Consumer method), [4](#)
validate() (user.User method), [40](#)