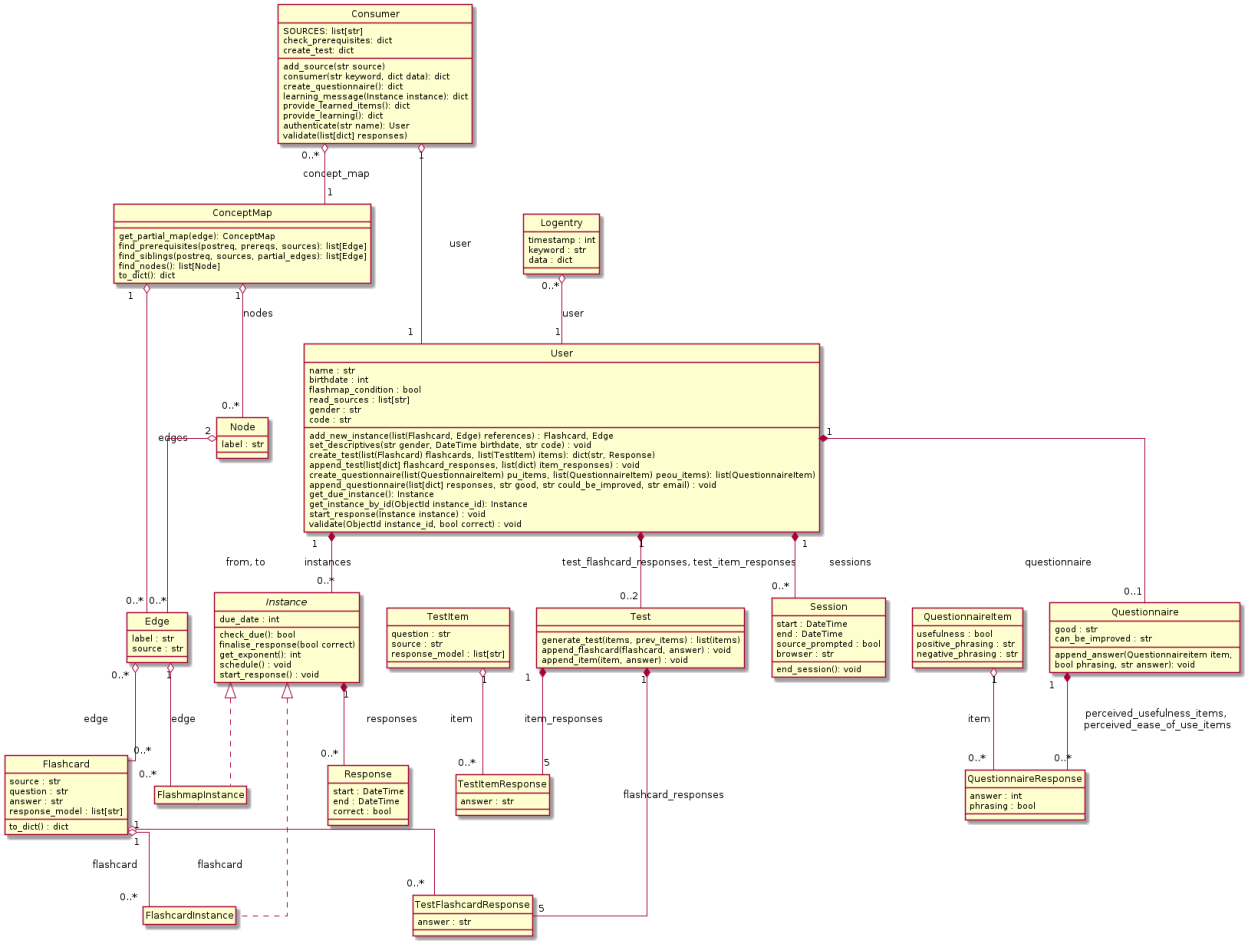
Flashmap server Documentation

Release 1.0

M.C. van den Enk

May 09, 2017

CONTENTS


Consumer
SOURCES: list[str]
check_prerequisites: dict
create_test: dict
add_source(str source)
consumer(str keyword, dict data): dict
create_questionnaire(): dict
learning_message(Instance instance): dict
provide_learned_items(): dict
provide_learning(): dict
authenticate(str name): User
validate(list[dict] responses)
concept_map
ConceptMap
get_partial_map(edge): ConceptMap
find_prerequisites(postreq, prereqs, sources): list[Edge]
find_siblings(postreq, sources, partial_edges): list[Edge]
find_nodes(): list[Node]
to_dict(): dict
user
Logentry
timestamp : int
keyword : str
data : dict
nodes
edges
Node
label : str
User
name : str
birthdate : int
flashmap_condition : bool
read_sources : list[str]
gender : str
code : str
add_new_instance(list[Flashcard, Edge] references) : Flashcard, Edge
set_descriptives(str gender, DateTime birthdate, str code) : void
create_test(list[Flashcard] flashcards, list[TestItem] items): dict(str, Response)
append_test(list[dict] flashcard_responses, list(dict) item_responses) : void
create_questionnaire(list[QuestionnaireItem] pu_items, list[QuestionnaireItem] peou_items): list(QuestionnaireItem)
append_questionnaire(list[dict] responses, str good, str could_be_improved, str email) : void
get_due_instance(): Instance
get_instance_by_id(ObjectId instance_id): Instance
start_response(Instance instance) : void
validate(ObjectId instance_id, bool correct) : void
from, to
instances
test_flashcard_responses, test_item_responses
sessions
questionnaire
Instance
due_date : int
check_due(): bool
finalise_response(bool correct)
get_exponent(): int
schedule() : void
start_response() : void
Edge
label : str
source : str
TestItem
question : str
source : str
response_model : list[str]
Test
generate_test(items, prev_items) : list(items)
append_flashcard(flashcard, answer) : void
append_item(item, answer) : void
Session
start : DateTime
end : DateTime
source_prompted : bool
browser : str
end_session(): void
QuestionnaireItem
usefulness : bool
positive_phrasing : str
negative_phrasing : str
Questionnaire
good : str
can_be_improved : str
append_answer(QuestionnaireItem item, bool phrasing, str answer): void
edge
edge
responses
item
item_responses
item
perceived_usefulness_items, perceived_ease_of_use_items
Flashcard
source : str
question : str
answer : str
response_model : list[str]
to_dict() : dict
FlashmapInstance
Response
start : DateTime
end : DateTime
correct : bool
TestItemResponse
answer : str
QuestionnaireResponse
answer : int
phrasing : bool
flashcard_responses
flashcard
flashcard
FlashcardInstance
TestFlashcardResponse
answer : str

CHAPTER

ONE

CONTENTS:

1.1 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 self.nodes given a list of self.edges

Parameters **self.edges** (*list(Edge)*) – The list of self.edges for which to find the self.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 self.edges given a certain edge from a list of self.edges, filtered by a list of sources

Parameters

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

Returns A list of self.edges which are prerequisites from edge

Return type list(*edge*)

find_siblings(*edge*, *sources*, *partial_edges*)
Return a list of self.edges which are siblings of the given edge and have the same label

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 self.edges for exclusion when looking for siblings

Returns A list of edges which are siblings of edge and have the same label

Return type list(*edge*)

get_partial_map(*edge*, *sources*)
Returns a concept map containing only the parent and sibling self.edges together with the referred self.nodes

Parameters

- **edge** (Edge) – The input edge
- **sources** (*list (string)*) – The list of sources to filter on

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

Return type *ConceptMap*

to_dict()
Returns a dictionary representation of this object

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

Result The dictionary representation, compatible with visjs

Return type dict

1.2 controller module

class controller.**Controller**(*database*)
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

- **database** – The mongodb to connect to
- ***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

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 self.username

check_prerequisites()
Checks whether the self.user still has to fill in forms and returns the appropriate message

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

Return type dict

controller(*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)

learning_message(*item*)
Generates a learning message for the provided instance

Parameters **instance** (Instance) – The instance which has to be rehearsed

Returns The message with keyword "LEARNING RESPONSE" and data containing the partial concept map or flashcard dict representation

Return type dict

provide_learned_items()
Provides an overview of all learning

Returns A partial concept map containing all instances for this self.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

1.3 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
- **sources** – The source where this edge is described (e.g. paragraph 13.2 from Laagland)

to_dict()
Returns a dictionary representation of this object

It contains an 'id', 'label', 'from', 'to', and 'sources' entry

Returns The dictionary representation of this object, compatible with visjs

Return type dict

1.4 flashcard module

class flashcard.**Flashcard**(**args*, ***values*)
Bases: mongoengine.document.Document

A class representing a flashcard

Variables

- **question** – The question on the front side of the flashcard
- **answer** – The answer on the back side of the flashcard
- **sources** – The edges where this flashcard is based on
- **response_model** – A list consisting of parts of valid responses to the question (for the test matrix)

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

1.5 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

1.6 flashmap_instance module

class flashmap_instance.**FlashmapInstance**(**args*, ***kwargs*)
Bases: *instance.Instance*, mongoengine.document.EmbeddedDocument

A class for storing responses from the flashmap system

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

1.7 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 controller() 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

1.8 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

1.9 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

1.10 node module

class node.**Node**(**args*, ***values*)
Bases: mongoengine.document.Document

A class for representing nodes in the concept map

Variables **label** – The label appearing within the node

to_dict()
Returns a dictionary representation of this object

It contains an 'id' and 'label' entry

Returns The dictionary representation of this object, compatible with visjs

Return type dict

1.11 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

1.12 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

to_dict(*phrasing*)
A method for generating a dictionary representation of this object

Parameters **phrasing** (*boolean*) – Whether the positive or negative question is required

Returns The representation containing an id field, a phrasing field and a question field

Return type dict

1.13 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

1.14 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)

1.15 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

1.16 test module

class test.**Test**(**args*, ***kwargs*)
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

append_flashcard(*flashcard*, *answer*)
Adds a flashcard response to this test

Parameters

- **flashcard** (Flashcard) – The flashcard this item refers to
- **answer** (*string*) – The answer to the flashcard provided by the user

append_item(*item*, *answer*)
Adds an item response to this test

Parameters

- **item** – The test item this item refers to
- **answer** (*string*) – The answer to the flashcard provided by the user

randomise(*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*)

1.17 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

1.18 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

to_dict()
A method for generating a dictionary representation of this object

Returns The representation containing an id field and a question field

Return type dict

1.19 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

1.20 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")
- **`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
- **`source_requests`** – The days that the user was prompted a source request
- **`successful_days`** – The days that the user successfuly completed a session
- **`briefed`** – Whether the user already got the briefing after the experiment

`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*

`add_source`(*source*)

Adds a read source to self

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

`append_questionnaire`(*responses*, *good*, *can_be_improved*, *email*)

A method for appending a questionnairy to the user given responses

Parameters

- **`responses`** (*list(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 = 'flashcard') and an answer (key = 'answer')
- **item_responses** (*dict*) – A list of dict objects containing a TestItem (key = 'item') and an answer (key = 'answer')

check_due(*item*)
Checks whether the provided item is due for review

Parameters **item** (*Edge or Flashcard*) – The item to which the checked instance refers to

create_questionnaire(*pu_items*, *peou_items*)
A method for creating a new questionnaire

Parameters

- **pu_items** – A list of perceived usefulness items
- **peou_items** – A list of perceived ease of use 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, *Flashcard* or *TestItem*)

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*

get_instance_by_id(*instance_id*)
Retrieves an instance based on a provided instance id

Parameters **instance_id** (*ObjectId*) – The id of the instance to be requested

Returns The instance or None if no instance with instance_id exists

Return type *Instance*

retrieve_recent_instance()
Retrieves the instance most recently answered by the user

Returns The instance with the latest response.end being the most recent of all instances

Return type *instance*

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

time_spend_today()
A method for calculating the amount of seconds the user has spend on practicing flashcards

Returns The amount of seconds between every start and end of all responses of all instances of today

Return type int

undo()
Removes the response last submitted by the user, reschedules the respective instance, and returns the referred flashcard or edge

Returns The flashcard or edge referred to by the instance with the latest response

Return type *Flashcard* or *Edge*

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

CHAPTER

TWO

INDICES AND TABLES

- genindex * modindex * search

PYTHON MODULE INDEX

INDEX

S

T

U

V