MOOC - Requirements

Please assign your team a particular functionality in the respective section.

# FEATURES

User Login / Logout - ***Team******Juggernauts***

User Registration - Dalet

List categories - ***Team Pinnacle***

Add categories

-------------- 1st group : Teams Pinnacle, Team SAS, Crazy 4, Dalet, MyyApps

Add course (automated)

Delete course

Update course

Grading

Quizzes

----------- 2nd group: Teams Horizon, Stoneage, Synergy, Dalet(1 more missing here)

Discussion forum - ***Team Dynamo***

Announcement

Drop Course

Enroll Course: Student Id, course Id are given when enrolled to a course.

----- 3rd group: Teams Dynamo, HeadStart (1 more missing here)

# 

# 

# 

# 

# 

# 

# 

# Each user will have two categories: Courses he has enrolled to and courses he has uploaded. Anyone can add a course and anyone can

# 

# User: Email, Password. First name Last name, Address, City.

# Category: CategoryName, CategoryId

# Course : Course Id, UserId (who uploads it), Title, description, attachment, Category Id Discussion forum (optional): Course Id

# Discussion: Text, Title

# Quiz: Questions(more than 1), 4 options per questions, 1 correct answer.

# Drop / Enroll course: Student Id, Course Id[ ]

# Announcement: Announcement Id, Title, CourseId.

# 

# 

# RESTful API *(This will be move to another doc later)*

(URLs here are only designed for RESTful Service, not for Web Pages)

- Proposed by Wilson (Yufan) Yang <[yyfearth@gmail.com](mailto:yyfearth@gmail.com)>

Use JSON as the format for the body of both request and response, except for login and view.

For all response, there should be a header: “Content-Type: application/json; charset=utf-8”

The RESTful API URL patterns is designed by following the recommendation from the Book: OReilly - Restful Web Services Cookbook (2010).

Some principals I followed:

1. Stateless - no session is stored
2. Represents as Resources/Services (Nouns) instead of Actions (Verbs)
3. Use single noun for resource deal with a single entity (e.g. create and search/list), use plural for a set of entities (e.g. get by ID, update and delete)
4. Using lowercase\_separated\_by\_underscores followed the naming convention of Ruby, Python and MongoDB
5. (not finished yet :)

Currently, the API is mainly designed for the server-to-server interaction, for clients, it should be some additional authentication and authorization needed. Basic Auth and OAuth should be introduced for Access control, but I think we do not have enough time for that.

## Login / Logout

### User Login

#### Usage

Use for login into system for both service provider and consumer(server or client).

POST email and password to login, or use GET without password to just display the login page.

A callback URL should given if it designed to go back to the consumer’s web page.

The service provider should create a session id and write to the cookie.

It is NOT a RESTful API, instead it an API for web interface, designed for the teams do not want to implement the login page in their own web server.

It should be implemented by the teams, who selected the user parts, in there web server.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body (Encoded Form)* |
| POST | http://host:port/login?callback=:**url** | email=**:email**&password=**:password** |
| GET | http://host:port/login?email=**:email**&callback=:**url** | - |

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Method |
| callback | The URL direct to after login successfully | GET (in URL) / POST (in Body) |
| email | The email of the user want to login | GET (in URL) / POST (in Body) |
| password | The original not hashed password | Must use POST (in Body) |

#### Response

It always show the login page, email should be already filled in if it is given.

If both the email and password are given via POST, the page should submit automatically.

After login succeed, it will redirect to the URL given in the callback parameter, if it is not given, just go to the default page after login.

If login failed, just go back to the login page and require the user login manually, but please preserve the callback URL, since it should work after a success login.

## 

### User Logout

#### Usage

Use for logout the system for both service provider and consumer(server or client).

Just call the logout URL, and it should be clear the session if user logged in.

A callback URL should given if it designed to go back to the consumer’s web page.

The service provider should clear the session id and the cookie.

It is NOT a RESTful API, instead it an API for web interface, designed for the teams do not want to implement the login page in their own web server.

It should be implemented by the teams, who selected the user parts, in there web server.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/logout?callback=:**url** | - |

##### Parameters

|  |  |
| --- | --- |
| Parameter | Value |
| callback | The URL direct to after logged out successfully |

#### Response

It should be the same behaviour as user logout using the provider's web GUI.

After logged out, it will redirect to the URL given in the callback parameter, if it is not given, just go to the default page after logged out.

If user is not logged in or any other errors, it should display the error information in a page as same as a user using the web GUI.

## User

### User Validation (optional)

#### Usage

This is a programmatic way to check the user’s authentication, can be used for User Login in server-side only, but not equal to login.

Currently, [Basic Auth](http://en.wikipedia.org/wiki/Basic_access_authentication) (email + password) should be supported.

For Basic Auth: Use the user’s email as its “username” and user SHA-1 hashed password as its “password”.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/user/validation | (Authorization Header Needed) |

#### Response

|  |  |  |
| --- | --- | --- |
| [*Status Code*](http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html) | *Response Body* | *Condition* |
| 200 (OK) | Success JSON | Success |
| 400 (Bad Request) | Error JSON | Email is invalid |
| 404 (Not Found) | Error JSON | User not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Get User by Email

#### Useage

It used for any situation when servers or clients need the user’s information by given email.

DO NOT use it for User Login, since it is not require a password and not return with password.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/user/**:email** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | User JSON **w/o password** | Success |
| 400 (Bad Request) | Error JSON | Email is invalid |
| 404 (Not Found) | Error JSON | User not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### View User by Email

#### Useage

#### It will redirect to the specified user profile’s web view of the service provider.

It redirect to a HTML web page instead of return the JSON data. (Login may required)

It is designed for teams depend on the service but not implement the web GUI themselves.

It can achieved by URL rewrite.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/user/**:email**/view | - |

#### Response

It should redirect to the URL of the web view for the specified user profile using 301 (Moved Permanently) with URL in Location Header.

If there is any error, display the error page instead.

### Search Users

### Create User

#### Usage

Create a new user for user registration.

The password should be always hashed by SHA-1, lower-cased hex string.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| POST | http://host:port/rest/user**s** | User JSON with password |

##### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 201 (Created) | User JSON | Success |
| 409 (Conflict) | Error JSON | Email is duplicated |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Save/Update User

#### Usage

Can be used for both creating and updating user, since the email address is unique.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| PUT | http://host:port/rest/user/**:email** | User JSON \* |

#### \*Password is required for creating new user, but it will be ignored for updating.

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 201 (Created) | User JSON **w/o password** | Create new user if not exists |
| 200 (OK) | User JSON **w/o password** | Update existing user |
| 400 (Bad Request) | Error JSON | Email or JSON is invalid |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Update User Password

#### Usage

Since update user do not contain password, use this to change user’s password.

The password should be always hashed by SHA-1, lower-cased hex string.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| PUT | http://host:port/rest/user/**:email**/password | The hashed password |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | User JSON **w/o password** | Update successful |
| 400 (Bad Request) | Error JSON | Email or password is invalid |
| 404 (Not Found) | Error JSON | User not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Delete User

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| DELETE | http://host:port/rest/user/**:email** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Success JSON | Delete success |
| 400 (Bad Request) | Error JSON | Email is invalid |
| 404 (Not Found) | Error JSON | User not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

## Category

### Get Category by ID

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/category/**:id** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Category JSON | Success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | Category not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### View Category by ID

#### Useage

#### It will redirect to the specified category’s web view (may show the courses) of the service provider.

It redirect to a HTML web page instead of return the JSON data. (Login may required)

It is designed for teams depend on the service but not implement the web GUI themselves.

It can achieved by URL rewrite.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/category/**:id**/view | - |

#### Response

It should redirect to the URL of the web view for the specified category using 301 (Moved Permanently) with URL in Location Header.

If there is any error, display the error page instead.

### 

### List All Categories

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/categor**ies** | - |

##### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Category Array JSON | Success |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Add Category

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| POST | http://host:port/rest/categor**ies** | Category JSON **w/o ID** |

##### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 201 (Created) | Category JSON **w/ ID** | Success |
| 409 (Conflict) | Error JSON | Name is duplicated |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Update Category

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| PUT | http://host:port/rest/category/**:id** | Category JSON |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Category JSON | Update success |
| 400 (Bad Request) | Error JSON | ID or JSON is invalid |
| 404 (Not Found) | Error JSON | Category not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Delete Category

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| DELETE | http://host:port/rest/category/**:id** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Success JSON | Delete success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | User not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

## Course

### Get Course by ID

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/course/**:id** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Course JSON | Success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | Course not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### View Course by ID

#### Useage

#### It will redirect to the course home page of the service provider.

It redirect to a HTML web page instead of return the JSON data. (Login may required)

It is designed for teams depend on the service but not implement the web GUI themselves.

It can achieved by URL rewrite.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/course/**:id**/view | - |

#### Response

It should redirect to the URL of the specified course home page using 301 (Moved Permanently) with URL in Location Header.

If there is any error, display the error page instead.

### 

### Search Courses

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/course**s**?param=value&... | - |

##### Parameters

All parameters are optional, usually are specified by default. If no parameter is given, return all.

|  |  |
| --- | --- |
| q | A part of title, description, category, etc. (like google search) |
| title | Exactly the title |
| category\_id | Exactly the category |
| created\_by | Exactly the creator’s email |
| created\_from/to | The time span for created\_at property |
| updated\_from/to | The time span for updated\_at property, can be used for Sync |
| offset | For paging, the offset index of entity for this return, default 0 |
| limit | For paging, the numbers of entity for this return, default is unlimited? |
| order\_by | For ordering and paging, default is order by id |

##### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Course Array JSON | Success |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### View Course List

#### Useage

#### It will redirect to the course list page of the service provider.

It redirect to a HTML web page instead of return the JSON data. (Login may required)

It is designed for teams depend on the service but not implement the web GUI themselves.

It can achieved by URL rewrite.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/course**s**/view | - |

#### Response

It should redirect to the URL of the course list page using 301 (Moved Permanently) with URL in Location Header.

If there is any error, display the error page instead.

### 

### Add Course

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| POST | http://host:port/rest/course**s** | Course JSON **w/o ID** |

##### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 201 (Created) | Course JSON **w/ ID** | Success |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Update Course

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| PUT | http://host:port/rest/course/**:id** | Course JSON |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Course JSON | Update success |
| 400 (Bad Request) | Error JSON | ID or JSON is invalid |
| 404 (Not Found) | Error JSON | Course not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Delete Course

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| DELETE | http://host:port/rest/course/**:id** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Success JSON | Delete success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | User not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### List All Participants

#### Usage

A shortcut for getting all participants’ user profile, to avoid fetching the participants one by one.

It will return all user entities **without additionally attribute like role**.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/course/**:id**/participants | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | User Array JSON **w/o password** | Success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | Course not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### View Participants List

#### Useage

#### It will redirect to the specified participants web view of the service provider.

It redirect to a HTML web page instead of return the JSON data. (Login may required)

It is designed for teams depend on the service but not implement the web GUI themselves.

It can achieved by URL rewrite.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/course/**:id**/participants/view | - |

#### Response

It should redirect to the URL of the web view for the specified user profile using 301 (Moved Permanently) with URL in Location Header.

If there is any error, display the error page instead.

### 

### Drop / Enroll Course

Just update the course entity with course id by adding or removing a user from the participants.

## 

## Discussion

### Get Discussion by ID

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/discussion/**:id** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Discussion JSON | Success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | Discussion not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### View Discussion by ID

#### Useage

#### It will redirect to the specified discussion web view, of the service provider, show messages.

It redirect to a HTML web page instead of return the JSON data. (Login may required)

It is designed for teams depend on the service but not implement the web GUI themselves.

It can achieved by URL rewrite.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/discussion/**:id**/view | - |

#### Response

It should redirect to the URL of the web view for the specified discussion using 301 (Moved Permanently) with URL in Location Header.

If there is any error, display the error page instead.

### View Discussion List

#### Useage

#### It will redirect to the discussion list web view of the service provider.

It redirect to a HTML web page instead of return the JSON data. (Login may required)

It is designed for teams depend on the service but not implement the web GUI themselves.

It can achieved by URL rewrite.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/discussion**s**/view | - |

#### Response

It should redirect to the URL of the web view for the discussion list using 301 (Moved Permanently) with URL in Location Header.

If there is any error, display the error page instead.

### Search Discussions

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/discussion**s**?param=value&... | - |

##### Parameters

All parameters are optional, usually are specified by default. If no parameter is given, return all.

|  |  |
| --- | --- |
| q | A part of title, description, course name (like google search) |
| course\_id | Exactly the course id (to support discussion without course\_id, use “none”) |
| title | Exactly the title |
| created\_by | Exactly the creator’s email |
| created\_from/to | The time span for created\_at property |
| updated\_from/to | The time span for updated\_at property, can be used for Sync |
| offset | For paging, the offset index of entity for this return, default 0 |
| limit | For paging, the numbers of entity for this return, default is unlimited? |
| order\_by | For ordering and paging, default is order by id |

##### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Discussion Array JSON | Success |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Add Discussion

#### Request

|  |  |  |
| --- | --- | --- |
| Method | URL | Request Body |
| POST | http://host:port/rest/discussion**s** | Discussion JSON **w/o ID** |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 201 (Created) | Discussion JSON **w/ ID** | Success |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Update Discussion

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| PUT | http://host:port/rest/discussion/**:id** | Discussion JSON |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Discussion JSON | Update success |
| 400 (Bad Request) | Error JSON | ID or JSON is invalid |
| 404 (Not Found) | Error JSON | Discussion not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Delete Discussion

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| DELETE | http://host:port/rest/discussion/**:id** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Success JSON | Delete success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | User not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Get Messages of a Discussion

#### Usage

Get all messages of a discussion with paging and delta sync support.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/discussion/**:id**/messages?params... | - |

##### Parameters

All parameters are optional, usually are specified by default. If no parameter is given, return all.

|  |  |
| --- | --- |
| created\_from/to | The time span for created\_at property |
| updated\_from/to | The time span for updated\_at property, can be used for Sync |
| offset | For paging, the offset index of entity for this return, default 0 |
| limit | For paging, the numbers of entity for this return, default is unlimited? |
| order\_by | For ordering and paging, default is order by id or created\_at? |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Message Array JSON | Success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | Discussion not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Get Message from a Discussion by ID

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/discussion/**:discussion\_id**/message/**:message\_id** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Message JSON | Success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | Discussion not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Add Message to a Discussion

#### Request

|  |  |  |
| --- | --- | --- |
| Method | URL | Request Body |
| POST | http://host:port/rest/discussion**/:discussion\_id**/message**s** | Message JSON **w/o ID** |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 201 (Created) | Message JSON **w/ ID** | Success |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Update Message of a Discussion

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| PUT | http://host:port/rest/discussion/**:discussion\_id**/message/**:message\_id** | Message JSON |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Message JSON | Update success |
| 400 (Bad Request) | Error JSON | ID or JSON is invalid |
| 404 (Not Found) | Error JSON | Discussion not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Delete Message from a Discussion

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| DELETE | http://host:port/rest/discussion/**:discussion\_id**/message/**:message\_id** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Success JSON | Delete success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | User not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

## Quiz

### Get Quiz by ID

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/quiz/**:id** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Quiz JSON | Success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | Quiz not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### View Quiz by ID

#### Useage

#### It will redirect to the specified quiz web view of the service provider.

It redirect to a HTML web page instead of return the JSON data. (Login may required)

It is designed for teams depend on the service but not implement the web GUI themselves.

It can achieved by URL rewrite.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/quiz/**:id**/view | - |

#### Response

It should redirect to the URL of the web view for the specified quiz using 301 (Moved Permanently) with URL in Location Header.

If there is any error, display the error page instead.

### Search Quizzes by Course ID

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/quizze**s**?course\_id=**:course\_id**  http://host:port/rest/course/**:course\_id**/quizze**s** | - |

##### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Quiz Array JSON | Success |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Add Quiz

#### Request

|  |  |  |
| --- | --- | --- |
| Method | URL | Request Body |
| POST | http://host:port/rest/quiz**zes** | Quiz JSON **w/o ID** |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 201 (Created) | Quiz JSON **w/ ID** | Success |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Update Quiz

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| PUT | http://host:port/rest/quiz/**:id** | Quiz JSON |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Quiz JSON | Update success |
| 400 (Bad Request) | Error JSON | ID or JSON is invalid |
| 404 (Not Found) | Error JSON | Quiz not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Delete Quiz

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| DELETE | http://host:port/rest/quiz/**:id** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Success JSON | Delete success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | User not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Get Submissions of a Quiz

#### Usage

Get submissions of a quiz with filter, paging and delta sync support.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/quiz/**:id**/submissions?params... | - |

##### Parameters

All parameters are optional, usually are specified by default. If no parameter is given, return all.

|  |  |
| --- | --- |
| created\_by | Filter by the student’s email (need?) |
| created\_from/to | The time span for created\_at property |
| updated\_from/to | The time span for updated\_at property, can be used for Sync |
| offset | For paging, the offset index of entity for this return, default 0 |
| limit | For paging, the numbers of entity for this return, default is unlimited? |
| order\_by | For ordering and paging, default is order by id or created\_at? |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Submission Array JSON | Success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | Quiz not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### View Submission List

#### Useage

#### It will redirect to the the submission list page of specified quiz of the service provider.

It redirect to a HTML web page instead of return the JSON data. (Login may required)

It is designed for teams depend on the service but not implement the web GUI themselves.

It can achieved by URL rewrite.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/quiz/**:id**/submissions/view | - |

#### Response

It should redirect to the URL of the web view for the submission list of specified quiz using 301 (Moved Permanently) with URL in Location Header.

If there is any error, display the error page instead.

### Get Submission of a Quiz by ID

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/quiz/**:quiz\_id**/submission/**:submission\_id** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Submission JSON | Success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | Quiz not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### View Submission

#### Useage

#### It will redirect to the specified submission web view of the service provider.

It redirect to a HTML web page instead of return the JSON data. (Login may required)

It is designed for teams depend on the service but not implement the web GUI themselves.

It can achieved by URL rewrite.

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| GET | http://host:port/rest/discussion/**:quiz\_id**/submission/**:submission\_id**/view | - |

#### Response

It should redirect to the URL of the web view for the specified submission using 301 (Moved Permanently) with URL in Location Header.

If there is any error, display the error page instead.

### Add Submission for a Quiz

#### Request

|  |  |  |
| --- | --- | --- |
| Method | URL | Request Body |
| POST | http://host:port/rest/quiz**/:quiz\_id**/submission**s** | Submission JSON **w/o ID** |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 201 (Created) | Submission JSON **w/ ID** | Success |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### 

### Update Submission for a Quiz

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| PUT | http://host:port/rest/quiz/**:quiz\_id**/submission/**:submission\_id** | Submission JSON \* |

The score will be ignored, since it designed for student if allowed, and other cannot change that.

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Submission JSON | Update success |
| 400 (Bad Request) | Error JSON | ID or JSON is invalid |
| 404 (Not Found) | Error JSON | Quiz not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Grading Submission for a Quiz

#### Usage

Since the update submission cannot update the score, this one is used for grading a submission

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| PUT | http://host:port/rest/quiz/**:quiz\_id**/submission/**:submission\_id**/score | The score value |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Success JSON | Update success |
| 400 (Bad Request) | Error JSON | ID or JSON is invalid |
| 404 (Not Found) | Error JSON | Quiz not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

### Delete Submission from a Quiz

#### Request

|  |  |  |
| --- | --- | --- |
| *Method* | *URL* | *Request Body* |
| DELETE | http://host:port/rest/quiz/**:quiz\_id**/submission/**:submission\_id** | - |

#### Response

|  |  |  |
| --- | --- | --- |
| *Status Code* | *Response Body* | *Condition* |
| 200 (OK) | Success JSON | Delete success |
| 400 (Bad Request) | Error JSON | ID is invalid |
| 404 (Not Found) | Error JSON | User not found |
| 500 (Internal Server Error) | Error JSON | Other Errors |

# 

# JSON Objects

The naming convention of properties is **lowercase\_separated\_by\_underscores** followed the same style of Ruby, Python and MongoDB.

*“Python and Ruby both recommend UpperCamelCase for class names, CAPITALIZED\_WITH\_UNDERSCORES for constants, and lowercase\_separated\_by\_underscores for other names.” (*[*Wikipedia*](http://en.wikipedia.org/wiki/Naming_conventions_(programming)#Python_and_Ruby)*)*

Every teams can extend these object for their own sake, but those extra properties may not be understandable by others.

## Success

For methods not return an Entity JSON, e.g. delete, return this to indicate the operation is done.

It can be detected by both status code 2xx and whether response\_json.ok exists.

{

ok: “OK”

message: “Category w/ ID ... is deleted” // the readable message

}

## Error

Errors can be detected by both status code 4xx/5xx and whether response\_json.error exists.

{

error: “USER\_NOT\_FOUND” // The error code

message: “Cannot find user with email [xxxxxx@xx.xxx](mailto:xxxxxx@xx.xxx)” // the readable error message

}

## User

{

email: (id)

password:

first\_name:

last\_name:

address: (optional) {

address:

city:

state:

zip: (optional)

country: (optional)

}

created\_at: ‘2013-04-18T08:56:20.583Z’ // ISO format

updated\_at: // default is the same as created\_at

}

## Category

{

id

name: // unique name

created\_at: ‘2013-04-18T08:56:20.583Z’ // ISO format (remove it if useless)

updated\_at: // default is the same as created\_at (remove it if useless)

}

## Course

{

id

title

description

attachments: []

category\_id: // or we can just use the category name as its id

~~user\_email: (who uploads it)~~

~~student\_emails: []~~

participants: [{

email: // user’s email

role: // student or instructor/owner or assistant

}, ...]

created\_by: // the user who create it, and auto add to the participants as instructor/owner

created\_at: ‘2013-04-18T08:56:20.583Z’ // ISO format

updated\_by: // user email (remove it if it is useless)

updated\_at: // default is the same as created\_at

}

## Discussion

{

id

course\_id: // optional, for discussion without course

title: // optional, the topic

created\_by: // user email

(messages: [] // array of Message, if your team choose to embed all messages)

created\_at: ‘2013-04-18T08:56:20.583Z’ // ISO format

updated\_at: // default is the same as created\_at

}

### Message

It is separated from Discussion to avoid the size grow too big and concurrency issue.

It depends on your team’s decision whether is should be embedded or not, but doubtful for large discussions.

{

title: // optional

content: // text

discussion\_id:

created\_by: // user email

created\_at: ‘2013-04-18T08:56:20.583Z’ // ISO format

updated\_at: // default is the same as created\_at

}

## Quiz

{

id

title: (optional)

description: (optional)

questions:[{ // Question 1..\*

content: // the content/description of the question

options: [] // 2..\*

answer: // the correct answer

}, ...]

created\_by: // user email

created\_at: ‘2013-04-18T08:56:20.583Z’ // ISO format

updated\_at: // default is the same as created\_at

graded\_by: // user email of who grading

graded\_at: // time of grading

}

### Submission

{

id

quiz\_id

score: // grading the submission for the quiz

answers: [0,1,2,...] // for each question, keep same sequence

created\_by: // user email

created\_at: ‘2013-04-18T08:56:20.583Z’ // ISO format

updated\_at: // default is the same as created\_at

}

## Announcement

{

id

title

content: // the text

course\_id

created\_by: // user email

created\_at: ‘2013-04-18T08:56:20.583Z’ // ISO format

updated\_at: // default is the same as created\_at

}