System API Design

# Module Registration API

The job of **Module Registration** API is to take a user and an academic year and return a list of modules. Admin users can also CRUD modules.

Consider use of API key in Authorisation header (font-end server will have a hard-coded API key).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Resource** | **GET**  read | **POST**  create | **PUT**  update | **DELETE** |
| /{year}/modules | Return a list of modules | Create a new module | Method Not Allowed (405) | Method Not Allowed (405) |
| /{year}/{module code} | Returns a specific module | Method Not Allowed (405) | Updates a specific module | Deletes a specific module |
| /{year}/modules?uid={} | Returns a list of modules from a given year that is associated with a specific user. | Method Not Allowed (405) | Method Not Allowed (405) | Method Not Allowed (405) |
| /modules?uid={} | Returns a list of modules that is associated with a specific user. | Method Not Allowed (405) | Method Not Allowed (405) | Method Not Allowed (405) |
| /{year}/{modulecode}/students.csv | Returns the CSV file | Method Not Allowed (405) | Adds/Overwrites the student list file | Method Not Allowed (405) |

# Notifications API Design

The **Notifications** application will provide an API to update user notification settings. The front-end server will use this API. The notifications component will also have to access the Message Store’s API to get messages to email.

Consider use of API key in Authorisation header (font-end server will have a hard-coded API key).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Resource** | **GET**  read | **POST**  create | **PUT**  update | **DELETE** |
| /notificationsettings | Method Not Allowed (405) | Create a new notification settings entry | Method Not Allowed (405) | Method Not Allowed (405) |
| /notificationsettings/{uid} | Returns a specific user’s notification settings | Method Not Allowed (405) | Updates a specific user’s notification settings | Deletes a specific user’s notification settings |

# Message Store API

The **Message Store** provides an API to the front-end server to let users get messages for a given group. And let users CRUD messages. (Some actions like delete are restricted and logic for this will need to be either in front-end or message store)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Resource** | **GET**  read | **POST**  create | **PUT**  update | **DELETE** |
| /messages?gid={} | Return a list of all messages from a specific group | Method Not Allowed (405) | Method Not Allowed (405) | Method Not Allowed (405) |
| /messages/{message\_id} | Returns a specific message | Method Not Allowed (405) | Updates a specific message | Deletes a specific message |
| /messages/search  With specific HTTP headers for to be used for search  Modules: <user\_modules>  SearchText: <text>  SearchFilter:  <module>|<group>|<year>|ALL | Returns a list of messages that match the search | Method Not Allowed (405) | Method Not Allowed (405) | Method Not Allowed (405) |
| /messages/daily/  With specific HTTP header for user  <uid> | Returns a list of Daily Summary for that user | Method Not Allowed (405) | Method Not Allowed (405) | Method Not Allowed (405) |
| /messages/mention |  |  |  |  |
| /messages/replies |  |  |  |  |

There is a lot more to consider here, haven’t even written down parts of API used by the notifications module.

https://medium.com/stories-from-the-lab/advanced-rest-api-design-754d1f254e87