# Modules

Router - request validation, router HTTP request to module

**auth.py** - authenticate users, issue token, validate token

**admin.py** - Add/remove/update user information

Project - Add new project, assign segments to users, permission ownership push/pull

Editor - Speech-enhanced editor, add functionality, project status

Collator - Combine segments, editor segments, push back to user,

Notes:

* A database can by SQL, pickle, dictionary whatever is best
* Requests
  + Sent from HTTP client
  + Sent via HTTP body
  + JSON format
* HTTP methods:
  + GET: get data no modification
  + PUT: create new resource
  + POST: update resource
  + DELETE: remove resource
  + OPTIONS: what does the server supported
* RestFUL
  + Must be stateless NO modules holding volatile data
* RestFUL URI
  + Operations based on specific URI
  + E.G. LOGIN: /wsgi/login/
  + E.G. STATUS /wsgi/status/username
* Errors
  + I make use of valid HTTP codes: <https://en.wikipedia.org/wiki/List_of_HTTP_status_codes>
* Ownership
  + Push/pull document ownership
  + Only one person can edit a document at a time
  + Many people can read any document
* SQLITE3
  + <http://zetcode.com/db/sqlitepythontutorial/>
* Storage
  + Python-GIT: can backup repo and track changes

## Router

|  |  |
| --- | --- |
| **Component** | **Purpose** |
| Validate client request | Make sure it is a valid request |
| Redirect request to module | Link request to module and pass request data to module |
| Load services from JSON config file | Use JSON config to describe the service details |
| Handle JSON, MULTIPART/FORM-DATA and GET binary | Most request data will use JSON objects but audio is uploaded using multipart/form-data and audio is retrieved using GET |

Notes:

* The server response should be quick

## **auth.py** module

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Purpose** | **Assumption/Prerequisite** | **Typical/Response** | **Request** |
| Token generation  **gen\_token()** | Generate a new token for a valid user (used by **UserAuth.login()**) |  | Unique string that can be sent over HTTP.  E.g.:  JSON: {“**token**” : “YWU3YTVmYzMtNTJlZC00ZGNkLWE1MjctYjIyYTRlNDkzNWM1”} | --INTERNAL--  URI: None (used by **UserAuth.login()**) |
| Token validation  **token\_auth(token, authdb)** | Tokens represent logged-in users. Each request (except for user login) must forward a ***token*** to validate the request. The provided token must be compared to tokens in the ***auth*** database (tokens table) to determine which logged-in user has made the request. |  | Will return the ***username*** of the logged-in user (according to the tokens table entry) or raise **httperrs. NotAuthorizedError** | --INTERNAL--  URI: None (used by ***all functions that require authentication*** against a logged-in user). |
| **UserAuth** Class**:** |  |  |  |  |
| User authentication  **UserAuth.login()** | Validate provided username, password and role | Admin has added this user to the database. See **Admin.add\_user()** | 200 Ok  JSON: {"token": "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1"}  ---  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/SERVICE/login  JSON { "**username**": "user", "**password**": "pass", “**role**”: “role”} |
| User logout  **UserAuth.logout()** | Remove token from token pool (logged-in tokens) | Must have a valid token. See **UserAuth.login()** | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI:  /wsgi/SERVICE/logout  JSON: {"**token**": "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1"} |
| Remove stale tokens | Each token has a lifespan and stale tokens should be removed. Currently a removal of all stale tokens happens at user login. A stale token is also removed when found during token authentication (if found but expired). |  | --INTERNAL-- | --INTERNAL-- |
| User force-logout  **UserAuth.logout2()** | Remove stale token. User must provide username and password. This method is used if token has not been removed during previous session. Can be used with login() to perform re-login. | The user is registered on the system. | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“**message**”:”...”} | HTTP POST  URI: /wsgi/SERVICE/logout2  JSON {"**username**": "user", "**password**": "pass"} |
| Change user’s password  **UserAuth.changepassword()** | Change password of current user. This option will always be available once user logged in | User must be registered and logged in. If password forgotten, ask admin to reset password - see Admin.update\_user() | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“**message**”:”...”} | HTTP POST  URI:  /wsgi/SERVICE/changepassword  JSON {“**token**”: “YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1”, “**password**” : “pass”} |
| Reset user’s password  **UserAuth.reset\_password()** | Reset user’s password to temporary value and send to registered email address.  TODO: still need to send to email, currently sends password back in request |  | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“**message**”:”...”}  ***TODO: will send new password to user’s email address*** | HTTP POST  URI:  /wsgi/SERVICE/resetpassword  JSON {“**username**”: “user”} |

Notes:

* SERVICE can be ‘projects’ , ‘project/admin’, ‘editor’ or ‘editor/admin’
* Currently using SQLite to implement an ***auth*** database which contains ***users*** and ***tokens*** tables.
* For password hashing we make use of ***python-bcrypt***
* Plain text password are salted and hashed on the server not on the client.
* Make use of 4xx client errors (Python exceptions implemented in **httperrs.py**)
  + 200 OK - everything is good
  + 401 Unauthorized - not a valid user
  + 400 Bad Request - malformed request

## **admin.py** module

The admin class has two associated auth databases:

* The first ***authdb*** does the authentication of admin requests
* The second ***target\_authdb*** is the database which is under administration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Purpose** | **Assumptions** | **Typical/Response** | **Request** |
| **Admin** Class**:** |  |  |  |  |
| Add user  **Admin.add\_user()** | Add new user to the ***target\_auth*** database. | Assume user does not exist | 200 Ok  401 Not Authorized  400 Bad Request  409 Conflict  500 Internal Server Error  JSON {“**message**”: “...”} | HTTP POST  URI: /wsgi/SERVICE/admin/adduser  JSON { “**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**username**” : “user”, “**name**” : “first”, “**surname**” : “second”, “**email**” : “[a@b.com](mailto:a@b.com)”, “**role**” : “project;editor”} |
| Delete user  **Admin.del\_user()** | Remove user from ***target\_auth*** database. | Assume user exists | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“**message**”: “...”} | HTTP POST  URI: /wsgi/SERVICE/admin/deleteuser  JSON { “**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**username**”: “user” } |
| Request User information  **Admin.get\_uinfo()** | Load user’s information from database and send to client using JSON object | Assume user exists | 200 Ok  JSON: {“name” : “first”, “surname” : “second”, “email” : “[a@b.com](mailto:a@b.com)”}  -----  401 Not Authorized  404 Not Found  500 Internal Server Error  JSON {“message”: “...”} | HTTP POST  URI: /wsgi/SERVICE/admin/userinfo  JSON {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**username**”: “user”} |
| Update user information  **Admin.update\_user()** | Save updated user’s information to user’s database. Extract information from JSON object | Assume user exists and requested field(s) is valid | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“message” : “...”} | HTTP POST  URI: /wsgi/SERVICE/admin/updateuser  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**username**” : “user”, “**name**” : “first”, “**surname**” : “second”, “**email**” : “a@b.com”, “**password**” : “pass”, “**role**”: “editor”} |
| Load users  **Admin.get\_users()** | Provide a list of usernames to project interface so project created can assign segments to users |  | 200 Ok  JSON: {“username1” : {“name” : “first”, “surname” : “second”, “email” : “[a@b.com](mailto:a@b.com)”}, “username2” : {}, ...}  ---  401 Not Authorized  500 Internal Server Error  JSON {“message” : “...”} | HTTP POST  URI: /wsgi/SERVICE/admin/loadusers  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1"} |

Notes:

* SERVICE could be “projects” or “editor”
* SQLite for authentication database
  + **Potential performance optimisation:** Do keep database in module memory -- ramdisk can speed up things: http://www.jamescoyle.net/how-to/943-create-a-ram-disk-in-linux
* Errors (Python exceptions implemented in **httperrs.py**)
  + 400 Bad Request - malformed request
  + 404 Not found - user not found
  + 409 Conflict - user already exists

## Project

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Purpose** | **Assumptions** | **Typical/Response** | **Request** |
| Create project | Create a new project |  | 200 Ok  JSON: {“projectid” :”p…”}  ---  400 Bad Request  401 Not Authorized  500 Internal Server Error  JSON {“**message**”:”...”} | HTTP POST  URI: /wsgi/projects/createproject  JSON: { “**token**” :"YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**projectname**” : “name”, “**category**” : “cat\_name”, “**projectmanager**” : “name\_of\_someone”} |
| Load project | Load a specific project | Project must exist -- see **create project** -- and be unlocked and not in `unrecoverable` error | 200 Ok  JSON {"**project**": {"**category**": "NCOP", "**projectname**": "masefield", "**year**": 2016}, “**tasks**” : [{“**taskid**” : “tid”, “**editor**” : “editor\_name”, “**start**” : “0.0”, “**end**” : “10.0”, “**language**” : “English”}, …]}  ---  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/projects/loadproject  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**projectid**” : “pid”} |
| Create tasks | Save project updates or crate tasks defined by user for a specific project | Project must exist -- see **create project** -- and be unlocked and not in `unrecoverable` error. Audio must have been uploaded. **Cannot be run after assign tasks.** To update assignees or toggle ownership after assignment use **update\_project**, also see **assign tasks** | 200 Ok  400 Bad Request  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/projects/createtasks  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**projectid**” : “pid”, "**tasks**": [{"**editor**" : "neil", "**speaker**" : "Zuma", "**start**" : 0.0, "**end**" : 2.0, "**language**" : "English"}, {"**editor**" : "gamer", "**speaker**" : "Mandela", "**start**" : 4.0, "**end**" : 5.154830, "**language**" : "Zulu"}],  “**project**”: {"**projectname**": "saved\_project"}} |
| Delete Project | Delete project owned by user | Project must exist. | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI:  /wsgi/projects/deleteproject  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**projectid**” : “pid”} |
| Upload audio | Project owner uploads an audio file to server and assigns the audio to the project | Project must exist. Overrides existing audio. **Fails if project assigned.** | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST multipart/form-data  URI: /wsgi/projects/uploadaudio  MULTIPART/FORM-DATA  Requires body parameters:  “**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1"  “**projectid**” : “pid”  “**filename**” : “audio.ogg”  “**file**“ : (actual file binary data) |
| Get audio | Return the project audio that is accessible via a HTTP GET | Audio must be uploaded first -- see **upload audio** | 200 Ok  Audio Binary Data in Ogg format  ---  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP GET  URI: /wsgi/projects/getaudio  PARAMS  “**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1"  “**projectid**” : “pid” |
| List projects | List projects belonging to user |  | 200 Ok  JSON {"**projects**": [{"**category**": "NCOP", "**username**": "user01", "**creation**": 1476087930.204226, "**creator**": "user01", "**projectid**": "p4e3913c457d243518c76c81cfa2d9769", "**errstatus**": null, "**audiodur**": null, "**year**": 2016, "**jobid**": null, "**assigned**": "N", "**projectname**": "masefield", "**audiofile**": null}]}  ---  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/projects/listprojects  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1"} |
| List created projects | List projects created by user |  | 200 Ok  JSON {"**projects**": [{"**category**": "NCOP", "**username**": "user01", "**creation**": 1476087930.204226, "**creator**": "user01", "**projectid**": "p4e3913c457d243518c76c81cfa2d9769", "**errstatus**": null, "**audiodur**": null, "**year**": 2016, "**jobid**": null, "**assigned**": "N", "**projectname**": "masefield", "**audiofile**": null}]}  ---  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/projects/listcreatedprojects  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1"} |
| List categories | List project categories defined in project configuration |  | 200 Ok  JSON {“**categories**” : [“cat1”, “cat2”, …]}  ---  401 Not Authorized  500 Internal Server Error  JSON {“message” : “...”} | HTTP POST  URI: /wsgi/projects/listcategories  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1"} |
| List languages | List languages present in configuration |  | 200 Ok  JSON {“**languages**” : [“lang1”, “lang2”, …]}  ---  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/projects/languages  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1"} |
| Load users | List all registered users by role |  | 200 Ok  JSON {"user03": {"**surname**": "Kleynhans", "**name**": "Neil", "**email**": "neil@organisation.org"}, "user02": {"**surname**": "Kleynhans", "**name**": "Neil", "**email**": "neil@organisation.org"}, "user01": {"**surname**": "Kleynhans", "**name**": "Neil", "**email**": "neil@organisation.org"}, "user00": {"**surname**": "Kleynhans", "**name**": "Neil", "**email**": "neil@organisation.org"}}  ---  401 Not Authorized  500 Internal Server Error  JSON {“message” : “...”} | HTTP POST  URI: /wsgi/projects/loadusers  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1"} |
| Diarize audio | Segment the audio automatically using speech tools  Speech server will pull data from `getaudio` URL and PUT the results to `putresult` URL | Audio for project must exist. Project must be unlocked and not in unrecoverable error and **NOT ASSIGNED**. Speech server must be accessible and data URLs visible to speech server. | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI:  /wsgi/projects/diarizeaudio  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**projectid**” : “pid”} |
| Assign tasks | Assign tasks to selected editors -- sets ownership and project state. Creates documents for editors, checks task table fields are complete. | Requires saving of projects to be run first -- which ensures tasks exist. See **save project**. | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI:  /wsgi/projects/assigntasks  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**projectid**” : “pid”, “**collator**” : “name\_of\_collator”:} |
| Update project | Update fields of project or tasks. These would be -- PROJECT: projectname, category, collator, projectstatus, projectmanager  TASKS: editor, language, speaker, editing  TASKS is not needed when updating project details only | Project must exist and can only be run after **assign tasks**. | 200 Ok  400 Bad Request  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI:  /wsgi/projects/updateproject  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**projectid**” : “pid”, “**project**” : {“**projectname**” : “pname”, “**category**” : “cat\_name”}, "**tasks**": [{"**taskid**": 2, "**editor**" : "daniel", "**speaker**" : "Mandela", "**language**" : "Guangdong hua", “**ownership**”: 0}]} |
| Unlock project | Unlock project and try and stop speech job if still running | Project exists. Speech job has locked the project or system failure during project update operation. | 200 Ok  400 Bad Request  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI:  /wsgi/projects/unlockproject  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**projectid**” : “pid”} |

## Editor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Purpose** | **Assumptions** | **Typical/Response** | **Request** |
| Load tasks | Load all tasks assigned to editor/collator  Tasks are split into two groups: editor and collator |  | 200 Ok  {"**collator**": "No projects have been created", "**editor**": [{"category": "NCOP", "textfile": "/mnt/stp/storage/usr000/2017/01/20/p5c4ffe78ca0a4f02bc28fc0cc13c2fb9/000/text", "end": 3.144158128225064, "language": "FSVOtAMoBH", "projectid": "p5c4ffe78ca0a4f02bc28fc0cc13c2fb9", "completed": null, "creation": 1484911322.0, "year": 2017, "modified": 1485174763.0, "jobid": null, "commitid": "095a6d8503caa6a2875822448ebec110e4a29458", "start": 0.0, "speaker": "knyEtERTpK", "editor": "eusr001", "taskid": 0, "projectname": "kVFWQGJAFM", "editing": "eusr001", "errstatus": null}  ---  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI:  /wsgi/editor/tasks  JSON {“**token**” : “"YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1"} |
| Get text | Load document associated with task | User must have access to task - not speech or error locked | 200 Ok  JSON: { “**text**” : “...” }  ---  401 Not Authorized  404 Not Found  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/editor/gettext  JSON {“**projectid**”: “p0f6eb99011264373afa33058aed1f0a3”, “**token**”: “ODdlMTU1MzItNjkwZS00NDFiLWI5MTYtZDNlMTEzOWI1ZGZk”, “**taskid**”: 1} |
| Get audio | Load audio data associated with project. | User must have access to task - not speech or error locked | 200 Ok  Binary audio data returned  ---  401 Not Authorized  404 Not Found  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP GET  URI: /wsgi/editor/getaudio  PARAMS “**projectid**”: “p0f6eb99011264373afa33058aed1f0a3”  “**token**”: “ODdlMTU1MzItNjkwZS00NDFiLWI5MTYtZDNlMTEzOWI1ZGZk”  “**taskid**”: 1 |
| Save text | Save user’s edits to the document and update the status | User must have access to task -- not speech job lock, has ownership and not error locked | 200 Ok  401 Not Authorized  404 Not Found  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/editor/savetext  JSON {“**projectid**”: “p0f6eb99011264373afa33058aed1f0a3”, “**token**”: “ODdlMTU1MzItNjkwZS00NDFiLWI5MTYtZDNlMTEzOWI1ZGZk”, “**taskid**”: 1, “**text**” : “string to save”} |
| diarize | Schedule a diarize service to perform task on audio and update the text document. This will lock the editing of the document | User must have access to task -- not speech job lock, has ownership and not error locked. **Will only run if text document empty.** | 200 Ok  400 Bad Request  401 Not Authorized  404 Not Found  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/editor/diarize  JSON {“**projectid**”: “p0f6eb99011264373afa33058aed1f0a3”, “**token**”: “ODdlMTU1MzItNjkwZS00NDFiLWI5MTYtZDNlMTEzOWI1ZGZk”, “**taskid**”: 1} |
| recognize | Schedule a recognize service to perform task on audio and update the text document. This will lock the editing of the document | User must have access to task -- not speech job lock, has ownership and not error locked. **Will only run on empty portions and portions must exists -- added manually or through diarize.** | 200 Ok  400 Bad Request  401 Not Authorized  404 Not Found  500 Internal Server Error  JSON {“**message**” : “...”} | URI: /wsgi/editor/recognize  JSON {“**projectid**”: “p0f6eb99011264373afa33058aed1f0a3”, “**token**”: “ODdlMTU1MzItNjkwZS00NDFiLWI5MTYtZDNlMTEzOWI1ZGZk”, “**taskid**”: 1} |
| align | Schedule a alignment service to perform task on audio and update the text document. This will lock the editing of the document | User must have access to task -- not speech job lock, has ownership and not error locked. **Will only run on NON-empty portions and portions must exists -- added manually with text added or through diarize with text added or recognize.** | 200 Ok  400 Bad Request  401 Not Authorized  404 Not Found  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/editor/align  JSON {“**projectid**”: “p0f6eb99011264373afa33058aed1f0a3”, “**token**”: “ODdlMTU1MzItNjkwZS00NDFiLWI5MTYtZDNlMTEzOWI1ZGZk”, “**taskid**”: 1} |
| Unlock task | Cancel previously scheduled speech job and unlocks the task. Clears error status if in error | Speech was scheduled for task and must have ownership | 200 Ok  401 Not Authorized  404 Not Found  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/editor/unlocktask  JSON {“**projectid**”: “p0f6eb99011264373afa33058aed1f0a3”, “**token**”: “ODdlMTU1MzItNjkwZS00NDFiLWI5MTYtZDNlMTEzOWI1ZGZk”, “**taskid**”: 1} |
| Task done | Assigns ownership to collator | User has ownership of task, not locked | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/editor/taskdone  JSON {“**projectid**”: “p0f6eb99011264373afa33058aed1f0a3”, “**token**”: “ODdlMTU1MzItNjkwZS00NDFiLWI5MTYtZDNlMTEzOWI1ZGZk”, “**taskid**”: 1} |
| Load users | List all registered users by role |  | 200 Ok  JSON {"usr013": {"**surname**": "KtwciXjYmm", "**name**": "nebjD", "**email**": "BHjIV@plVbb.org"}, "usr018": {"**surname**": "dLfqRjgZPp", "**name**": "TABjL", "**email**": "[afeWI@LQkkK.org](mailto:afeWI@LQkkK.org)"}}  ---  401 Not Authorized  500 Internal Server Error | HTTP POST  URI: /wsgi/editor/loadusers  JSON: {“**token**” : "YmVkNWEyNzYtM2IwZS00ZDFmLTg0YjAtYzk0YjU3ZjI2N2I1", “**role**” : “editor”} |
| Clear Task Error | Clear error state of task |  | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“**message**” : “...”} | HTTP POST  URI: /wsgi/editor/clearerror  JSON {“**projectid**”: “p0f6eb99011264373afa33058aed1f0a3”, “**token**”: “ODdlMTU1MzItNjkwZS00NDFiLWI5MTYtZDNlMTEzOWI1ZGZk”, “**taskid**”: 1} |

Notes

* Use previous modified CKEditor
* Need to fix issues i.e. changing font size removes tags

## Collator

A collator inherits all the editor functionality and adds the following functions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Purpose** | **Assumptions** | **Typical/Response** | **Request** |
| Reassign tasks | Reassign the task back to the editor for fixes | The editor has marked the task as done |  | HTTP POST  URI: /wsgi/editor/reassigntasks  JSON {“**projectid**”: “p0f6eb99011264373afa33058aed1f0a3”, “**token**”: “ODdlMTU1MzItNjkwZS00NDFiLWI5MTYtZDNlMTEzOWI1ZGZk”, “**taskid**”: 1} |
| Build master | Collate all the task texts and create a single MS-WORD docx document. The document will be made available on a temporary URL which the user can retrieve using a GET | Text files exists for all tasks - i.e. assign tasks has been run | 200 Ok  401 Not Authorized  500 Internal Server Error  JSON {“**url**” : “...”}  The document can be retrieved at: /wsgi/editor/temporary\_url | HTTP POST  URI: /wsgi/editor/buildmaster  JSON {“**projectid**”: “p0f6eb99011264373afa33058aed1f0a3”, “**token**”: “ODdlMTU1MzItNjkwZS00NDFiLWI5MTYtZDNlMTEzOWI1ZGZk”} |