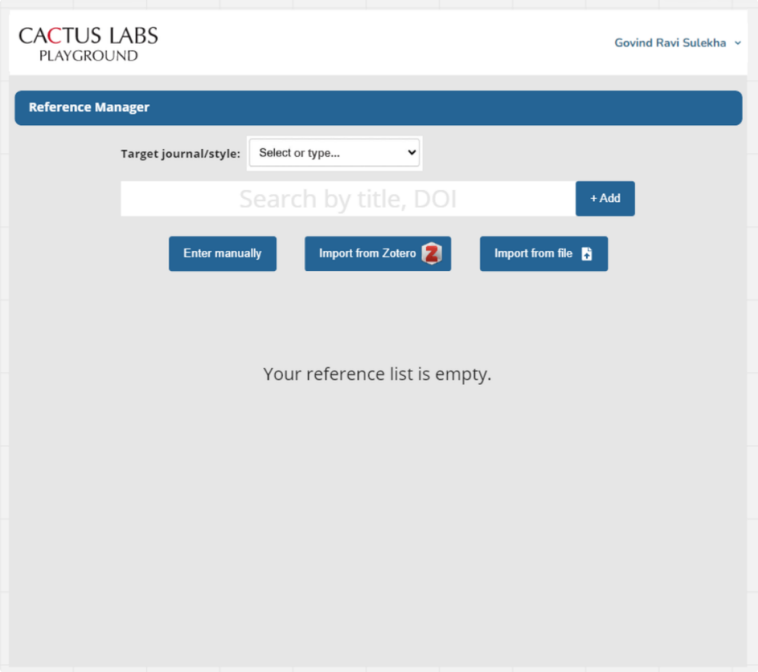
API Flow for Zotero users

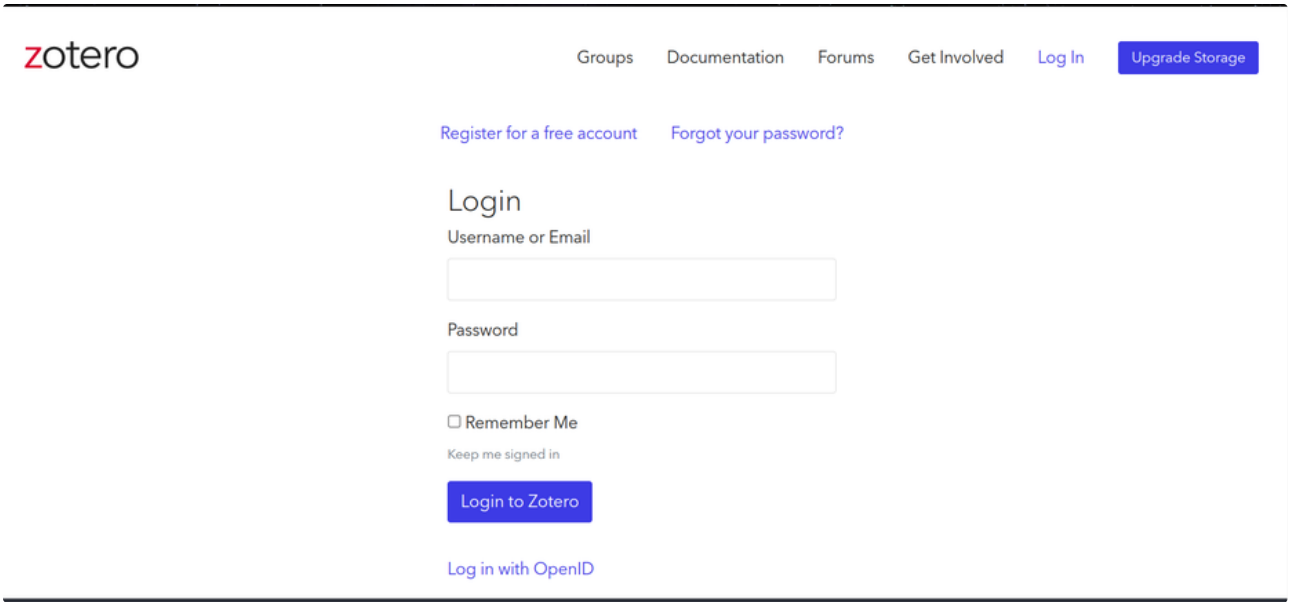
API 1: Login to Zotero and display the user's library items

Steps to be handled by Playground end for Zotero login

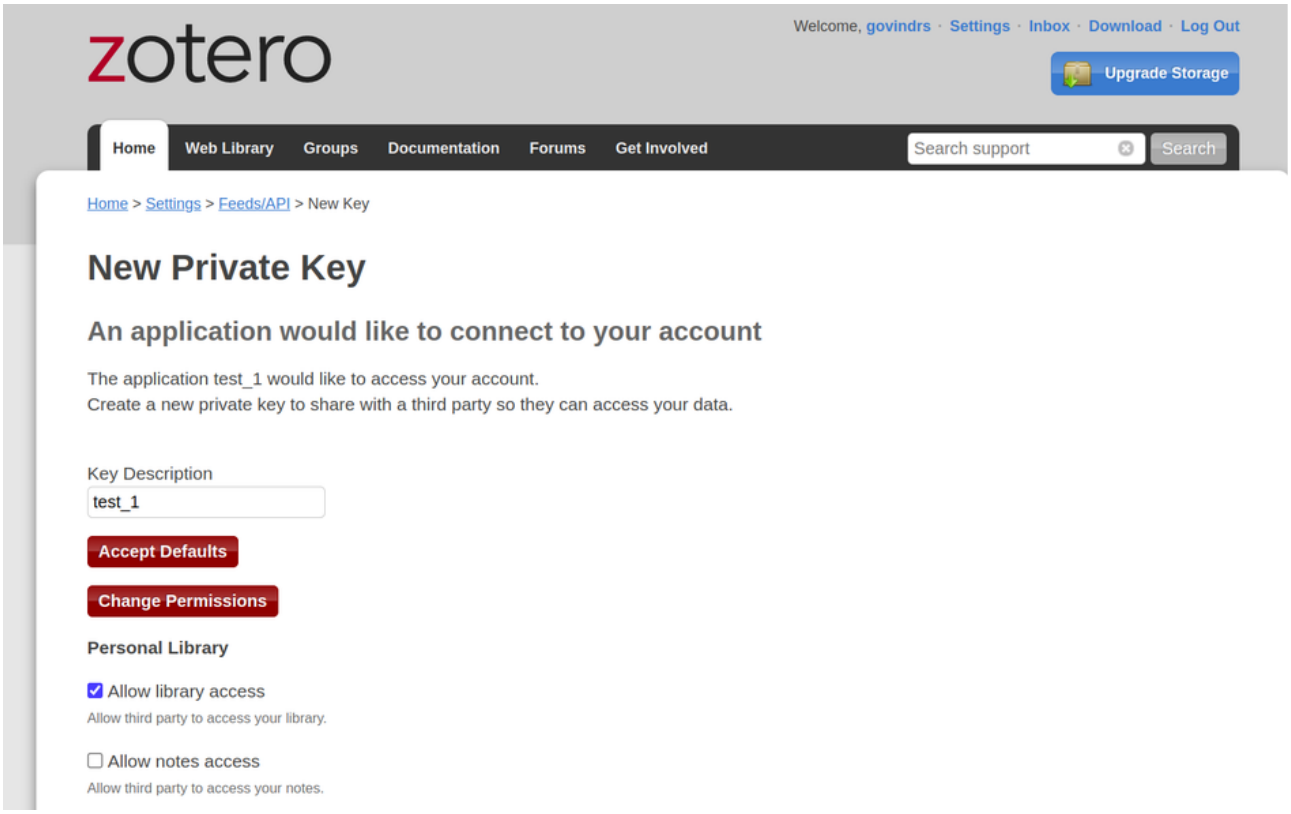
1. User comes to our platform and clicks on import from Zotero option



2. The user will be re directed to the Zotero login page and logs in to the account

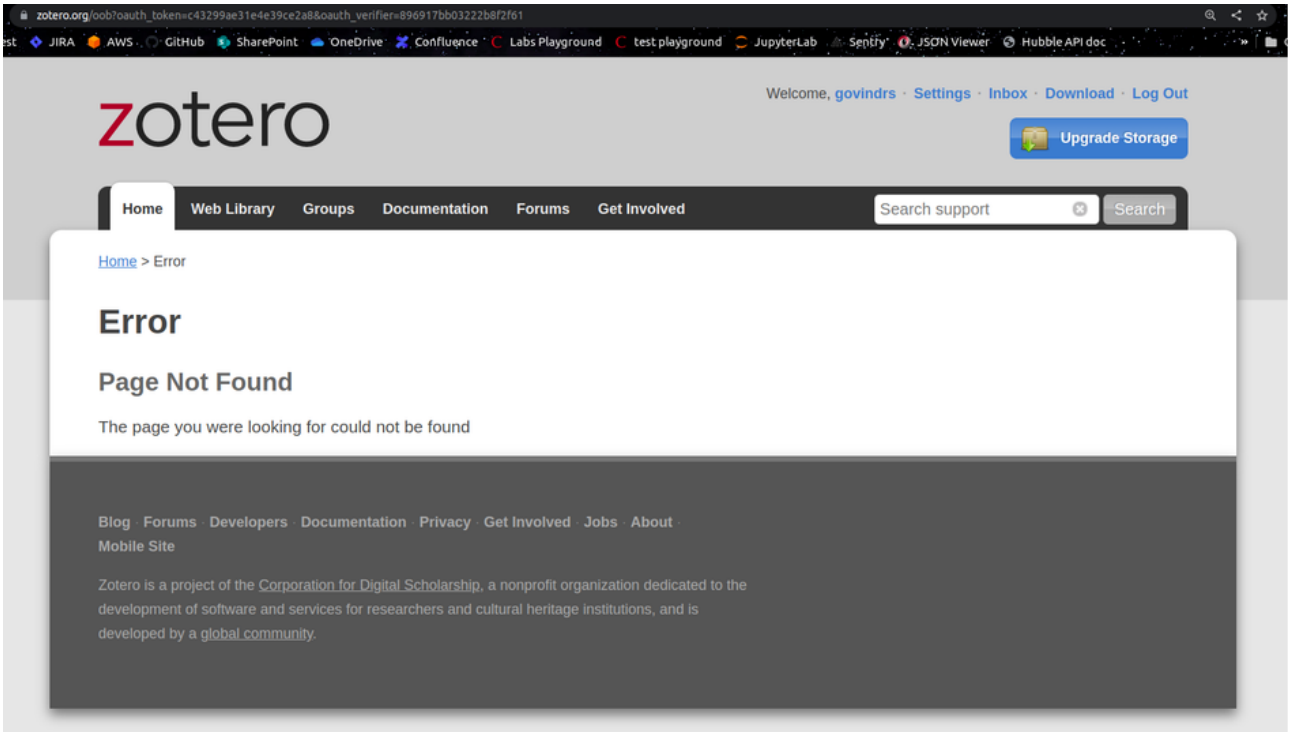


3. Now the user will be directed to the create API key page, where the user will create a new API key with the required permissions to access their zotero library and save it.



4. Now, the user currently is redirected to the Zotero page only with an error “Page not found”. Here we receive the callback URL, from which the Oauth token is extracted and used to authenticate the user.

NOTE: Playground will need to configure the redirection to our platform/ PG page.



5. Now once the user is authenticated Playground will hit the below API sending the required request_payload

   API endpoint : BASE URL + ' /import_zotero_items'

   Method: 'GET'

   Headers: *mandatory*

```
{
  "Content-Type": "application/json",
  "Authorization": "<static bearer token>"
}
```

   Request payload:
   We will fail the request if any of the below keys/ values are missing.

```
{
  "user_type": "zotero"
  "user_data": {
      "usre_id": "11222345",
      "user_name": "govindrs",
      "api_key": "<api_key>""
  },
}
```

NOTE:

- The API keys created for each user have no expiry
- Playground will need to store the below details in the DB, so that if the same user comes to our platform we do not have to perform the authentication step again for getting access to their zotero account and creating a new API key to access their library items
  - user_name
  - user_id
  - api_key

Backend flow to fetch the Zotero Library items and list them

1. Using the api_key and user_id from the request_payload we will hit the zotero API to get all the items.

```
zot = zotero.Zotero(userID, 'user', apiKey)
items = zot.everything(zot.top())
```

Sample response body:

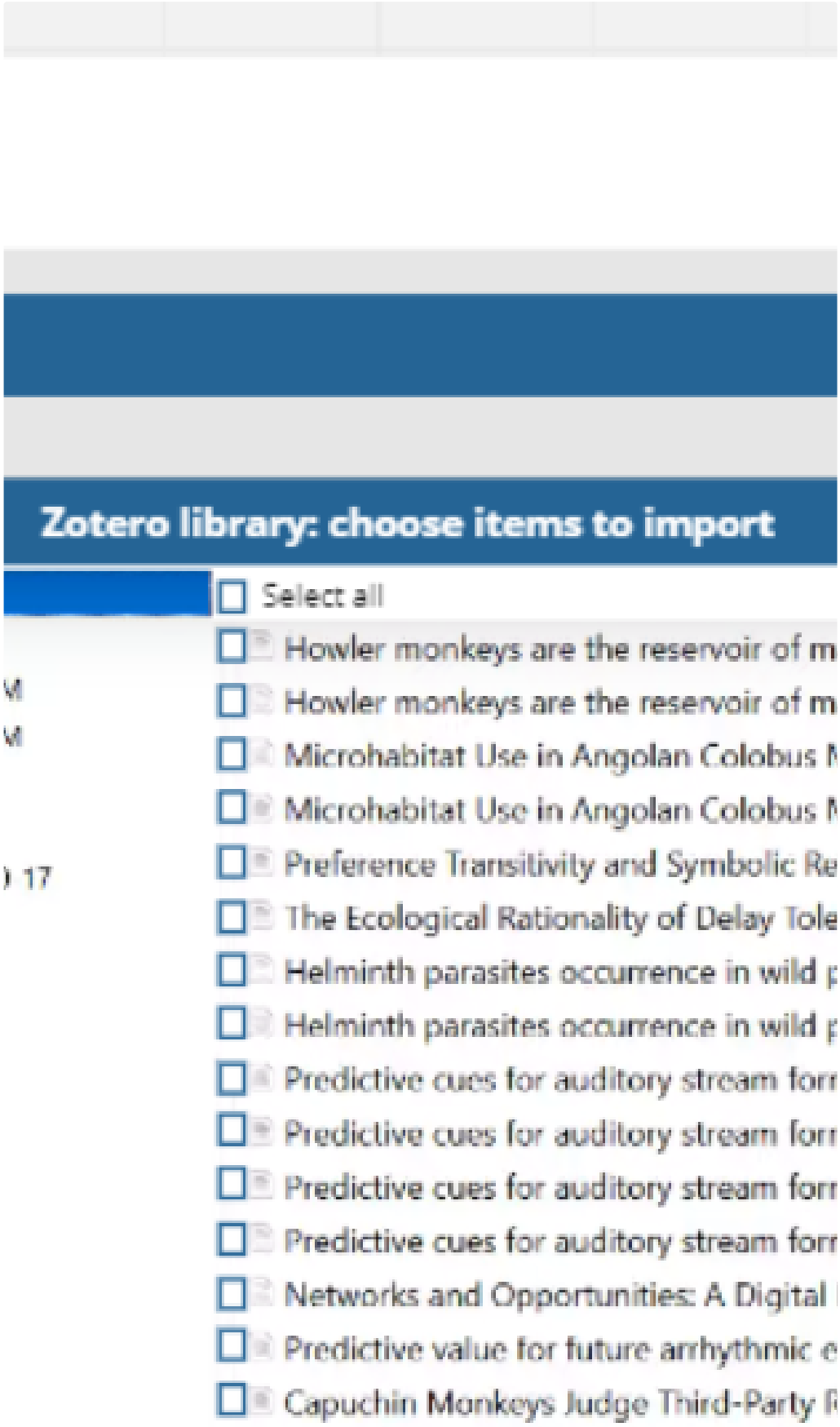
```
{
  "status": "success",
  "request_id": "",
  "user_type": "zotero",
  "zotero_items_metadata":[
      {
        'collection_name': "",
        'key': 'EKCTMWTI',
        'title': '',
        'collections': [
            'K6N3FA5L'
          ]
      },
      {
        'collection_name': "",
        'key': '123456',
        "title": ''
        'collections': [
            'K6N3FA5L'
          ]
      },
  ]
}
```

NOTE:

- When we fetch the items from Zotero, we will only send the required information needed to display on the Playground like the title and the collection name to make the required sub folders
- The remaining metadata (author_name, date, version, pages, etc.) about each item will be stored in Redis on our end only for 24 hrs.

API 2: Generate Citations for the selected items

Steps to be handled by Playground for getting the citations for selected item/items from the list



Now based on the above list of items displayed, the user will select one or multiple items and based on that the Playground will hit the below API passing the required request_payload

API endpoint : BASE URL + ' /get_citations'

Method: 'GET'

Headers: *mandatory*

```
{
  "Content-Type": "application/json",
  "Authorization": "<static bearer token>"
}
```

Request payload:

```
{
  "citation_style": "",
  "user_data": {
        "usre_id": "11222345",
        "user_name": "govindrs",
        "api_key": "<api_key>""
    },
  "selected_items": [
      {
        'key': 'EKCTMWTI',
      {
        'key': '123456',
```

```
      }
  ]
}
```

Backend flow to generate citations for the selected item/items

- From the request_payload we will parse the selected item key value and fetch the required data from Redis to generate an intermediate json_input for citeproc to get the citations in the required style.

```
citeproc_input_json = '''
[
    {
        "author": [
            {
                "family": "Schmidhuber",
                "given": "Jürgen"
            }
        ],
        "id": "ITEM-4",
        "issued": {
            "date-parts": [
                [2015]
            ]
        },
        "title": "Deep learning in neural networks: An overview",
        "container-title": "Neural Networks",
        "volume": "61",
        "page": "85-117",
        "type": "article-journal",
        "DOI": "10.1016/j.neunet.2014.09.003",
        "URL": "https://doi.org/10.1016%2Fj.neunet.2014.09.003",
        "publisher": "Elsevier {BV}"
    },
    {
        "id": "ITEM-5",
        "type": "book",
        "call-number": "SB419.5 .D85 2008",
        "edition": "Rev. and updated ed., [2nd ed.]",
        "event-place": "Portland, Or",
        "ISBN": "978-0-88192-911-9",
        "number-of-pages": "328",
        "publisher": "Timber Press",
        "publisher-place": "Portland, Or",
        "source": "Library of Congress ISBN",
        "title": "Planting green roofs and living walls",
        "author": [
            {
                "family": "Dunnett",
                "given": "Nigel"
            },
            {
                "family": "Kingsbury",
                "given": "Noël"
            }
        ],
```

```
        "issued": {
            "date-parts": [
                [
                    "2008"
                ]
            ]
    }
]
'''
```

- For displaying the Scite badge, DOI is required. We will also add a check to see if the DOI is present or not for a particular item. If the DOI is missing we wont fail the request, we will just send the below message

```
{
  "scite_badge": "DOI missing. No scite badge available"
}
```

Response Body

```
{
  'status': 'success',
  'request_id': '',
  'citation_style':'',
  'citations_for_item_selected': [
      {
          'key': '',
          'inline_citations': '',
          'full_citations': '',
          'scite_badge': ''
      },
      {
          'key': '',
          'inline_citations': '',
          'full_citations': '',
          'scite_badge': ''
      }
  ]
}

```

API 3: Edit the existing citations - TBD

- We will be deploying all API scripts to AWS lambda
- Citeproc style Files will be stored in the S3 bucket