With the addition of file sharing feature, the project admins and creators can add files and have the option to share the files with other creators in the project. If not shared, by default the uploaded file will be added as private files which are not visible to other creators.

API end points:

* UI to display resources to the user:
  + UI can be divided into 3 parts:
    - My private files: These are the files that the creator has not shared with other creators.
    - My shared files: These are the files that the creator has shared with other creators.
    - Shared files by other creators: These are the files that other creators has shared with the current user.
  + API:
    - Endpoint: /api/resource/project/{projectid}
    - GET request
    - Response:
      * 200 OK
      * Data: Resource []
    - Data response will be grouped by private files, my shared files, and shared files by others for ease of display.
* Create folder/Upload file:
  + End Point: /api/Resource
  + POST request
  + Request body:
    - Resource {   
      name = “Folder/File name”,   
      Owner = Person,   
      Project = Project,   
      isFolder = true/false (set based on if folder/file is being added),   
      Parent = Resource (this is optional, this can be set in case of nesting a folder or if a file is added to a folder. If null, the new folder/file will be added at root level)  
      }
* Update Resource:
  + End Point: /api/Resource/{ResourceId}
  + PUT request
  + Request body:
    - Resource object
    - To move the folder/file the parentId needs to be set to the move the folder/file under that Folder.
    - Upload process can be chunked in an upload service which async takes the chunks and uploads them to the server, this can be a optimization to improve the upload time.
* Download folder/file:
  + End Point: /api/Resource/{ResourceId}
  + GET request
  + Response:
    - The end point figures out if the resource being requested for is a folder or a file.
      * If the requested resource is a folder: Recursively get all the children pass them to the FileZip service is called and return the FileStream in the response.
      * If the requested resource is a file: Return the file stream int the response.
    - Download requests can also be chunked async manner and returned to the client for performance optimazions.
* Delete folder/file:
  + End Point: /api/Resource/{ResourceId}
  + DELETE request
    - Handles the disassociation of the resource with all the users its currently being shared with.
    - If the resource that is being deleted is a folder, this request needs to handle the case of recursively going through all the children and deleting them.

**File Sharing API**

* Share folder/file with Creator(s)
  + Step 1: Get all creators in a project
    - End Point: /api/Project/{projectid}/persons
    - GET request
    - Response: Persons []
    - This end point should be smart enough to show only those users who are not currently sharing the resource.
  + Step 2: After the user is selected, add the user to the resource
    - End Point: /api/SharedResource/
    - POST request
    - Request body:
      * SharedResource {

Resource = {Resource object of the resource that is being shared},

Person = Person[] (Person(s) object(s) with whom the resource is being shared with})

}

* Remove Shared folder/file permissions from Creator(s)
  + Step 1: Get all creators who have access to the resource
    - End Point: /api/SharedResource/{resourceId}
    - GET Request
    - Response: SharedResource
  + Step 2: Remove a creator
    - End point: /api/SharedResource/Resource/{resourceId}/Person/{PersonId}
    - Delete request

**Statistics Api**

* Project admin requests statistics on a project
  + End point: /api/Statistics/project/{id}
  + GET Request
  + Response: Statistics