**Start django server:**  
Go to the root of the directory (“finalProject” folder), open it in console and run the server with the command daphne -p 8000 finalProject.asgi:application

**Start express server:**

Run the index.js file in your IDE or open the folder containing the index.js file in the console and type ‘node index.js’ to run the server.

This is the schema for which I implemented the CRUD functionality:  
  
name: String,

creationDT: Date,

lastModifiedDT: Date,

lastAccessedDT: Date,

size: Number,

description: String,

It’s a simple object containing basic metadata about a file.

In the django project, I created a new app called “fileStorage”, which is responsible for the file storage functionality of the project. It provides a set of endpoints to access the functionality:

* <http://127.0.0.1:8000/fs/viewFiles/>
* <http://127.0.0.1:8000/fs/uploadFiles/>
* <http://127.0.0.1:8000/fs/updateMetadata/><str:fileName>/
* http://127.0.0.1:8000/fs/viewFiles/deleteFile/<str:fileName>/
* <http://127.0.0.1:8000/fs/success/>

All of these pages are linked, so I suggest just visiting the uploadFiles/ endpoint to start.

I fill go over each of these endpoints and explain its functionality. I will touch on the endpoints in the Express project later in the document.

* <http://127.0.0.1:8000/fs/viewFiles/> endpoint:

This endpoint renders a form where the user can input a name, description and a file. The name field will contain the name under which the file will be saved. The description field is meant for the user to leave any remarks for the file being uploaded. Finally, the user can upload a file of their choosing. The file will be processed and saved under the new name and the file metadata along with the name and description will be saved in mongoDB via the express endpoint ‘<http://localhost:3500/md/new>’. Json data containing the metadata info contained in the schema is attached to the request, and the express

app processes the file metadata and saves it in mongoDB.

* <http://127.0.0.1:8000/fs/viewFiles/> endpoint:  
  This endpoint displays the names of all the files located in the file storage directory of the django project. Next to the name of each file displayed is the description of the file along with 2 buttons: edit and delete. Upon clicking the ‘edit’ button, you will be taken to the updateMetadata/ endpoint (described below). Upon clicking the ‘delete’ button, the django endpoint the appropriate file will be deleted, along with it’s metadata. The metadata of the file is deleted via the django endpoint <http://127.0.0.1:8000/fs/viewFiles/deleteFile/><str:fileName>/,afterwards <http://localhost:3500/md/deleteByName/>:name this express endpoint finds the file metadata in the database via the file name and then deletes it.
* [http://127.0.0.1:8000/fs/updateMetadata](http://127.0.0.1:8000/fs/updateMetadata/)/ endpoint:

This endpoint allows the user to edit the metadata of the file. When this endpoint is visited, a template is rendered, where all the metadata of the file is displayed. The user can change the description of the file on this page, the other metadata is changed by interacting with the file directly (for example, the last accessed time is going to change the next time you access the file). This endpoint updates the metadata of the file by access the <http://localhost:3500/md/updateByName/:name> endpoint of the express project. This endpoint simply takes the new metadata from the webpage, finds the appropriate metadata object and updates it.

To display the metadata of the file prior to editing it, the endpoint makes a call to the express project endpoint <http://localhost:3500/md/getByName/:name>

* <http://127.0.0.1:8000/fs/success/> endpoint:  
  The user is redirected to this endpoint if their actions are successful.

The express project endpoints

* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)new
* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)list
* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)update/:id
* <http://localhost:3500/md/updateByName/:name>
* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)get/:id
* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)delete/:id
* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)deleteByName/:name
* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)getByName/:name

These endpoints are pretty simple and are quite self explanatory, thus they don’t need to be individually explained in detail.

* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)new - creates a new metadata object in mongoDB based on the JSON data passed to it in the request. The JSON data should match the schema shown above.
* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)list - lists all the metadata objects in mongoDB. User for testing purposes
* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)update/:id - allows the user to update the data of a metadata object specified by id based on the JSON data passed in the request. The JSON data should match the schema shown above. Used for testing purposes
* <http://localhost:3500/md/updateByName/:name> - allows the user to update the data of a metadata object specified by name based on the JSON data passed in the request. The JSON data should match the schema shown above.
* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)get/:id - retrieves the metadata object specified by id. Used for testing purposes.
* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)delete/:id - deletes the metadata object specified by id. Used for testing purposes.
* [http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)deleteByName/:name - deletes the metadata object specified by name.[http://localhost:3500/md/](http://localhost:3500/md/updateByName/:name)getByName/:name - retrieves the metadata object specified by name.