File Sharing API Server

Project setup instructions

* System requirements:

1. Database: MySql 5
2. Caching: Redis
3. NodeJs latest lts
4. All commands must be run in the project root dir

* Installation steps:

1. Clone the project from Github (Main branch)
   https://github.com/shahnCM/fileSharingApiServer.git
2. Create a database in MySql, naming 'file_sharing_app'
3. Rename the *example.env* to *.env*
4. Configure Environment variables if needed
5. To install node dependencies, run `npm install`
6. To migrate database table, run `npm run migration`
7. To start the project, Run `npm start`
8. To start the project with hot reloading, run `npm run dev`

* Setup verification & Test instructions:

1. Unit test can be run to verify project setup
2. This test will verify database and redis connectivity
3. This test covers file upload, download & delete functionality
4. For automated unit testing, run `npm test`

Feature Summary

1. End Points:

   a. **POST files/**
      This endpoint is used to upload new files. It should accept "multipart/form-data" requests and return a response in JSON format with the following attributes: "publicKey", "privateKey".
      The input field is named as "file"
   b. **GET files/:publicKey**
      This endpoint is used to download existing files. It should accept "publicKey" as a request parameter and return a response stream with a MIME type representing the actual file format.
   c. **DELETE files/:privateKey**
      This endpoint is used to remove existing files. It should accept "privateKey" as a request parameter and return a response in JSON format confirming the file removal.

2. Storage access functionality:

   a. File access functionality has been kept separate by using *Factory Design Pattern*.
   b. The default implementation works with local files located inside a root folder defined in the "FOLDER" environment variable.
   c. Other storage providers can be added through new Storage Factory implementation.

3. Rate limiting:

   This API Server configurable daily rate limiting for download and upload limits the network traffic from the same IP address. It can be configured from *.env* file with 2 variables.
   RATE_LIMIT_ALLOWED_REQUEST=4
   RATE_LIMIT_WINDOW_IN_MINUTE=1 #takes time in minute
   according to this configuration only *4* downloads and *4* uploads can be performed from the same ip address within *1* minute.

4. File cleanup job:

   The API Server has an internal job to cleanup uploaded files after configurable period of inactivity. The job frequency & file inactivity period both are configurable through the following *.env* variables.
   FILE_CLEAN_UP_INTERVAL=20 #takes time in second
   FILE_MAX_INACTIVE_TIME=60 #takes time in second
   according to this configuration the file clean up will run in every 20 seconds for each individual file and to delete the file when that file is inactive for 60 seconds (no downloads in 60 seconds).

5. Tests:

   All the HTTP REST API endpoints are covered by unite test as well as integration test in the scope of Database connection and Redis connection.
   To perform automated tests. Run `npm test` in project root dir