Share-IITK(Backend)

Abhay Pratap Singh
Parv Mor

June 2017

Team Members:

- Abhay Pratap Singh
- Parv Mor

Important Links:

- Share-IITK(Backend) repo

Timeline

1st week:

- Started learning scala from this book Functional Programming in Scala
- Jumped to this link Twitter Scala School, as I found it more useful.
- Came across Neophytes Guide to Scala, studied part - 8 from there, which was about concurrent side of scala.

2nd week:

- Learned about sbt build tool.
- Read about the features of REST API.
- Decided the endpoints for the API. There will be basically four endpoints:
  - /api/resources/search(GET): It will return the whole table(in JSON) to be searched in the frontend.
  - /api/resources/upload(POST): For uploading the respective file.
  - /api/resources/(GET): For downloading the file, distinguished by parameter MD5. Also, it will increment the score of that file by 1.

    - /api/courses/(GET): Return the list of courses.

- Learned about Actors in Akka and how they are used for asynchronous calls.

3rd week:

- Followed some links to get an idea of akka http REST API, for example:
    - Reactive REST Services using Akka HTTP
    - Building a REST Service in Scala with Akka HTTP, Akka Streams and Reactive Mongo - DZone Java
    - How to build a REST API with Akka Http
- Wrote the code for handling HttpResponses using akka for Share IITK repo.
- Started learning Slick(Scala Language-Integrated Connection Kit), which is Functional Relational Mapping (FRM) library for Scala that makes it easy to work with relational databases
- Decided to use flyway for Database migrations.

4th week:

- Read initial chapters from the book Essential Slick.
- Tried understanding code sample from Essential-Slick-Code
- Understood concept of DSL Routing partially and how it is used to respond to incoming HTTP requests. Implemented DSL routing instead of Function level interface for handling HTTP requests.
- Started writing code for uploading file using multi-part upload(as I found it more preferable after searching for hours on google).

5th week:

- Used slick for handling postgres queries.
- Wrote upload(without md5 checking), search endpoint.
- Implemented md5 checking in upload endpoint.
- Added download endpoint.

6th week:

- Implemented md5 checking in upload endpoint.
- Endpoints were working fine.

7th week:

- Scraped list of courses into a csv file from List of Courses,IIT KAnpur, using python(Beautiful Soup and requests).
- Made some minor changes in the upload endpoint to save the extension with the file.
- Tried deploying the API on Heroku, but was having some problem in using postgres on Heroku.
- API was not integrated with Frontend as per now.