Software Engineering: **G15**

A Self-hosted API for Web Metadata Retrieval

➔ Stakeholders and Users:

Here is the list of all the users and affected stakeholders of your project.

Stakeholders:

1) Developers:
   - Who contributes to this open-source project.
2) Companies/Developers:
   - Who integrates our service with their product (Web/Mobile App)
3) End-users:
   - Everyone who uses our service integrated products.
4) Donor/Sponsors:
   - Who monetarily supports our open-source project.

Users:

1) Our Service Integrated Products:
   - All the products (Web/Mobile App) that perform API calls to our service.
2) Companies/Developers:
   - Who registers on our website to use our services in their products.

➔ User Requirements:

1) User Registration:

- Anyone who wants to use our services has to register first using a valid email id.
- Successful registration generates a unique token (API_KEY) for each user, which will be later used to perform API Calls.

2) User Profile:

- User must be logged-in in order to access his profile.
- Profile shows User's **API_KEY** and **Remaining Daily API Call Limit**.
- Password-reset Option, In case the user wants to change his/her account's password.

3) API End-Point:

- Which can be used in Web/Mobile Apps to request for Web Page's Metadata.
- GET Method will be used to perform these API Requests.

Required Parameters to Perform API Call
String: API_KEY (Which was generated after successful registration)
String: WEBPAGE_URL (URL of the desired Web Page to retrieve Metadata)
Boolean: CACHED (Whether cached data will be provided as a response or not)

4) Documentation:

- Guide on how to use the service. So other developers can easily integrate the API in their products.
- It shows how to perform API calls from various frameworks (Java, JS, Python)
- Information about the Response of the API call.

➔ System Requirements:

- Active Internet Connection

- Hardware Requirements:
  - There won't be any specific hardware requirements to access the service. It can be integrated with any type of project. I.e. Mobile App, Web App, Desktop App.

- Server Requirements:

  - Minimum:
  Processor: 1.6GHz Dual-Core CPU
  RAM: 2GB
  HDD: 40GB
  OS: Ubuntu

  - Recommended:
  Processor: 1.6GHz Quad-Core CPU
  RAM: 4GB
  SSD: 50GB
  OS: Ubuntu

➔ Non-Functional Requirements:

1) The System should be reliable enough to respond to every request.
   → As we are going to make it Free to use service, Someone can break the server by calling the service inside an endless for loop!
   → The number of Requests for each user is limited to 1000 per day.

2) The System should provide high performance.
   → Caching will accomplish the Fast Response Time.
   → Thumbnail Compressor can be developed for bandwidth efficiency.

3) System maintenance:
   → Cache entry will be erased after 15 days.
   → If Cache memory reaches 85% of the size, then removal of data will start till memory reaches a safe state.

4) The System should be flexible enough to use on any platform.
   → API should be able to integrate into any iOS, Android, Web-based projects.

➔ User Interfaces:

1) Home Page:

   - There will be a Registration Section, Login Section, and a Button to access the Documentation.
   - Link for Github Repository of the project.

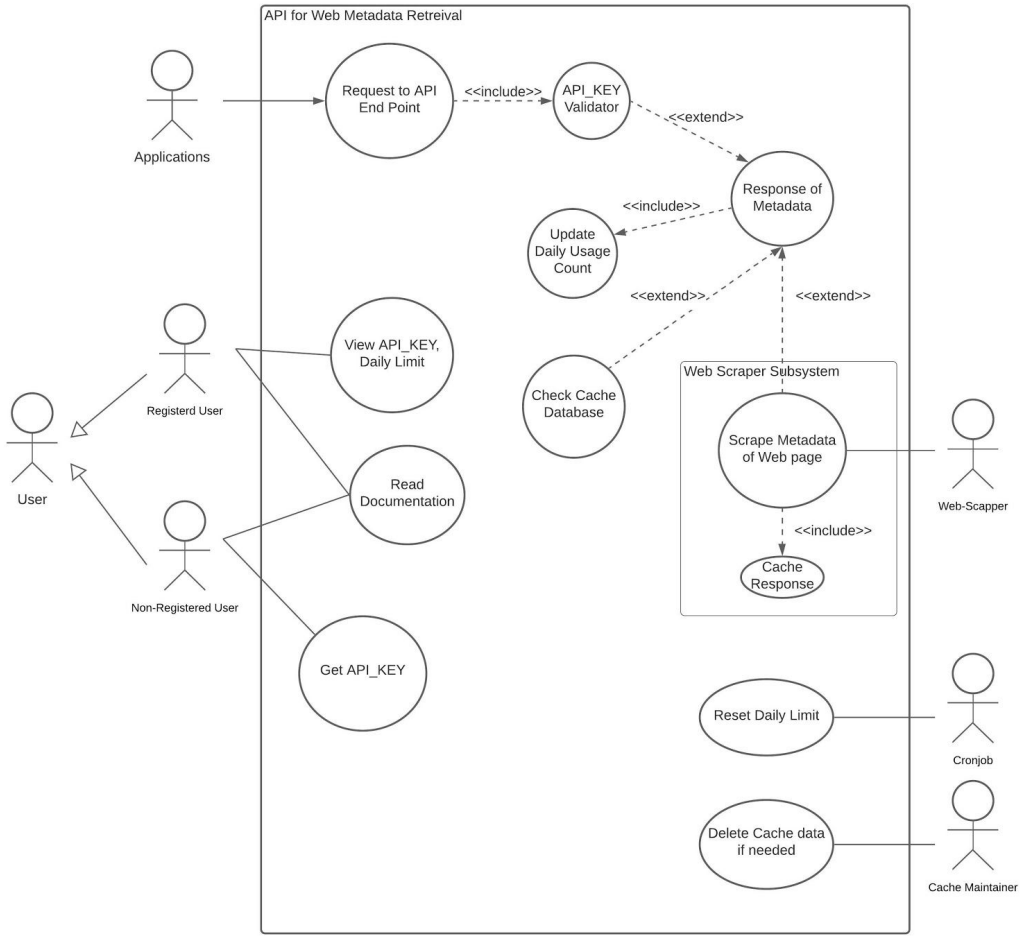
2) Profile Page:

   - User can see his/her API_KEY and remaining API calls limit on this page.
   - There will be buttons for password reset and logout.

➔ Open Issues:

- Private web-pages which can require some sort of authentication to access them, can't be scrapped. So we will not be able to retrieve metadata from such pages.
- If the HTML code of the requested web page won't be generic, there might be chances of wrong/missing metadata response.

➔ Use-Case Diagram:



Link: https://lucid.app/lucidchart/4e03d46a-0c5b-4769-8601-e7de94cd3c1d/edit?shared=true&page=0_0#

➔ Project Description:

Our Project is all about retrieval of Web Metadata. So our objective is to retrieve and present web metadata from a link that the user/developer provides during the API call.

Firstly, the users need to register themselves on our platform to use our API. After registration, an **API Key** would be generated that would be unique for every user. And to that API key, a specified number of usage limit would be added which in other words we call a daily limit for usage. After logging in, the users can access their API key, the daily limit left.

Users can call our API with any link of their choice that they want to retrieve metadata of and then our API will get to work. In the beginning, our API server will check for authenticity of the API_KEY and whether a user has exhausted his/her daily limit, if not then API will continue forward, or else it will return a response saying “Daily Limit Exhausted!”.

After that our service will check whether the metadata of the requested webpage is already available in the cache or not. If available and the request hasn’t asked for fresh scrap, the service will directly respond with cached data and will not perform the further steps.

API has the option for getting freshly scraped metadata if the user wants. For example, if a user wants to retrieve current metadata from the user-input webpage link then he/she can do that and in this case, our API won’t take metadata stored in the cache rather it will scrape metadata from that link again irrespective of whether the memory of metadata is present in the cache database.

The link for the requested web-page will be passed to our Web scraper. The scraper will scrape the metadata from the website if that website is not private. If the website is private then our API won’t be able to scrape metadata for it.

Then the metadata that our scraper has retrieved will be stored in the database cache and in case of duplicate values, old data will get replaced with fresh new data. And after that, the metadata will be sent to the requested client as a response.

Our API will also have a Cache Maintainer whose work would be to manage and clear cache. Whenever the user cache memory reaches 85% of a particular size, then removal of data will start till memory reaches a safe state or if it does not reach that level then it will be erased after 15 days of the initial entry.

Every time the users use our API, their daily limit will get reduced. And with the help of Cronjob, our API will reset the daily limit to the particular default limit at 12 AM IST daily, so that users don’t have any trouble using our API daily.