Software Engineering: **G15**

A Self-hosted API for Web Metadata Retrieval

---

1) **PROBLEM STATEMENT**

"A self hosted API for web metadata retrieval"

We are working towards making a service that returns metadata from a web page. If a user or web page wants to display data from a web page without linking to it or pasting all of its contents, this service can return a summary of information found on that URL with an appropriate thumbnail from it. The goal is to provide a free and open-source API that can be expanded to be used in mobile applications in the future. Our product intends to incorporate caching to allow for better performance.

2) **FUNCTIONAL REQUIREMENTS**

1. The system should be Free and Open Source.
   → The API will be free of cost for every user.
   → The Source Code of the system will be public on the Internet.

2. Account:
   → Developers have to register first in order to access the service.
   → We will provide API_KEY on successful registration.

3. API Endpoint:
   → Which can be used by other developers to access Web Metadata Retrieval Service.
   → GET Method will be used to perform API Requests.
   → Some of the tentative parameters for GET Request.

API_KEY
WEBPAGE_URL
CACHED

4. The system should return metadata of the requested web-page URL on every API call.

Response Attributes
Status Code
Title
Thumbnail
Page Content

5. The system should cache every response for better performance.

Database Table Attributes
Webpage Address (PK)
Title
Content
Thumbnail
Timestamp

6. Documentation:

   → We will provide a guide on how to use the service. So other developers can easily integrate the API in their products.

3) **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.

4) **ELICITATION TECHNIQUES**

We have used the following Elicitation Techniques:

- Observation and Analysis of existing system

We analyzed some of the popular social media networks, like Whatsapp, Telegram, Twitter and found similar functionality that our system aims for. We would like to replicate this feature as our system and also try to improve (if needed).

● Brainstorming :

Brainstorming is a technique for producing thoughts to take care of a structure issue. It includes a gathering, under the bearing of a facilitator. The quality of brainstorming is the potential members have in drawing associations between their thoughts in a free-thinking condition, in this way expanding the arrangement space.

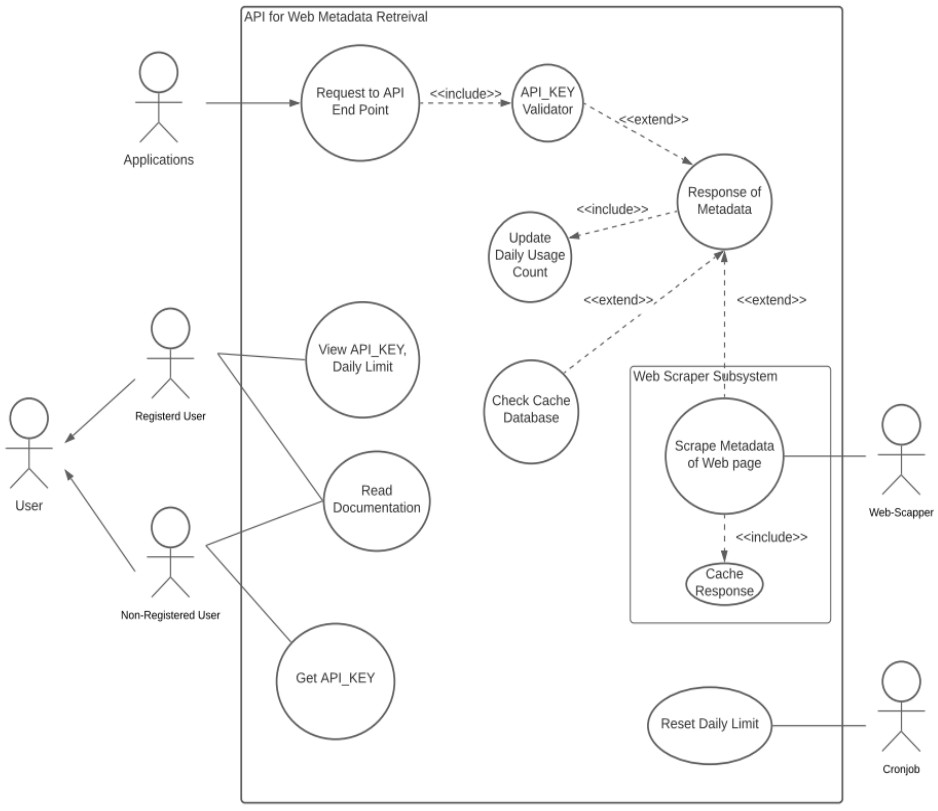
➔ **Storm Phase**

- ◆ The system should be free and open for contributions.
- ◆ Registration of users.
- ◆ Unique identification key for every user.
- ◆ Limit on the number of daily requests made by a user.
- ◆ The user should be able to view his\her remaining daily limit.
- ◆ Validate Webpage URL and also determine whether it’s private or public.
- ◆ Response of metadata retrieval scrapper should be cached.
- ◆ Send the actual thumbnail image in API Response.
- ◆ The system should provide an option to get cached or freshly scraped data.
- ◆ API response should have all the information about the webpage
- ◆ The system should be available 24*7.
- ◆ There must be proper documentation for API usage.
- ◆ The system should be flexible enough to support any platform.
- ◆ API Calls should have a low response time
- ◆ API Calls should be bandwidth-efficient

➔ **Final Phase**

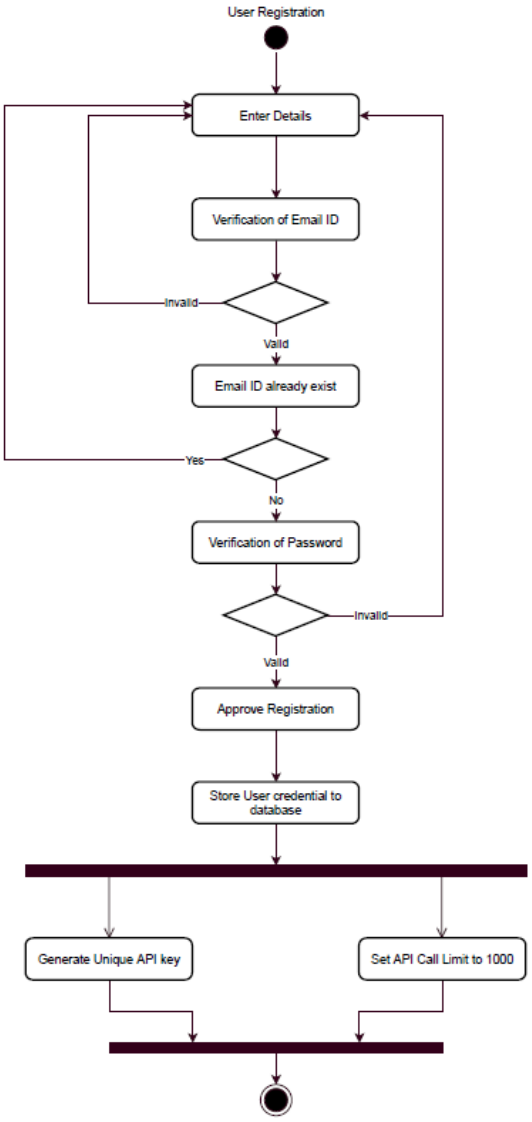
- Open Source
- User Registration/Login
- Unique Identification Key
- Daily Usage Limit and reset (Using Cron Job)
- User Profile Access
- Validation of URL
- Cache Management
- Option to choose response source (from cache or newly scraped)
- Availability
- Documentation
- Portability
- Performance

5) **USE CASE MODELS**

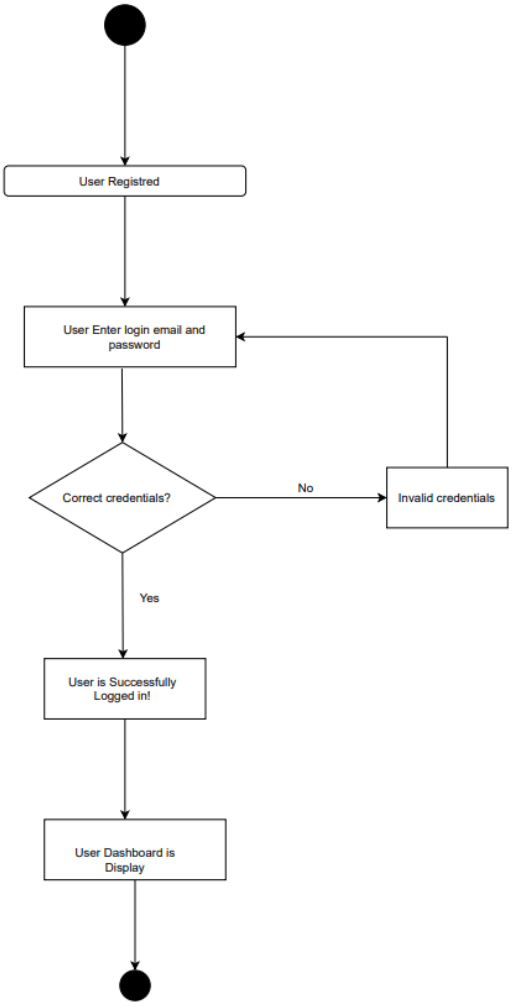


6) **ACTIVITY DIAGRAM**

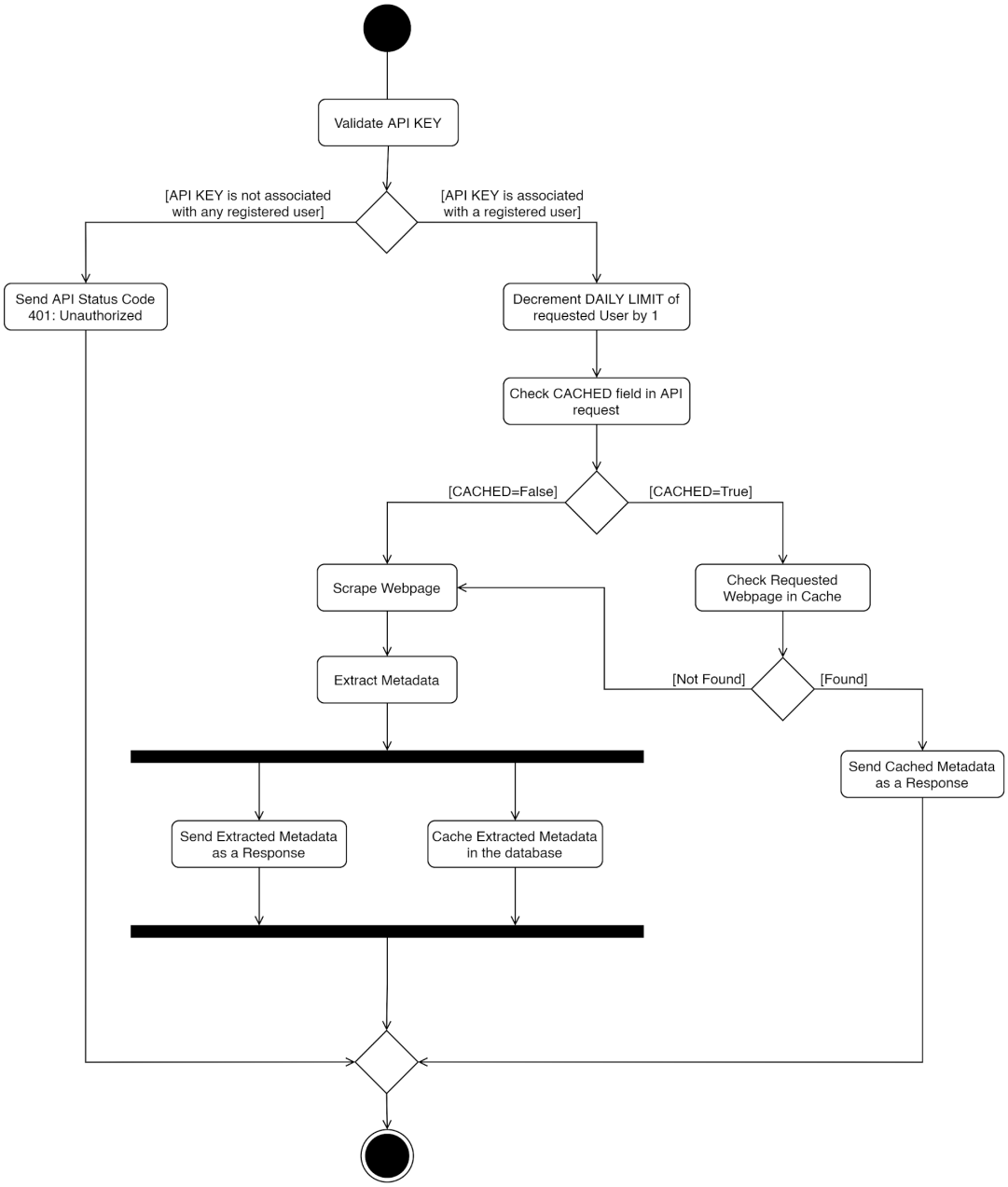
User Registration



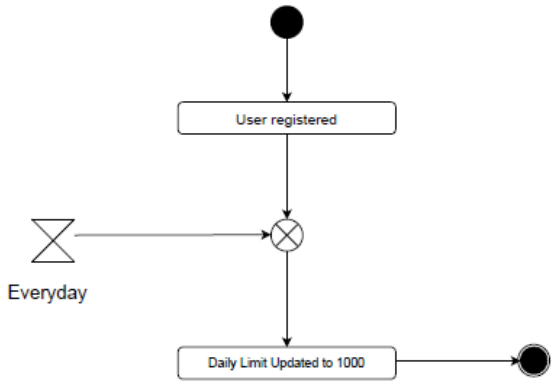
User Login


User Registred
User Enter login email and password
Correct credentials?
No
Invalid credentials
Yes
User is Successfully Logged in!
User Dashboard is Display

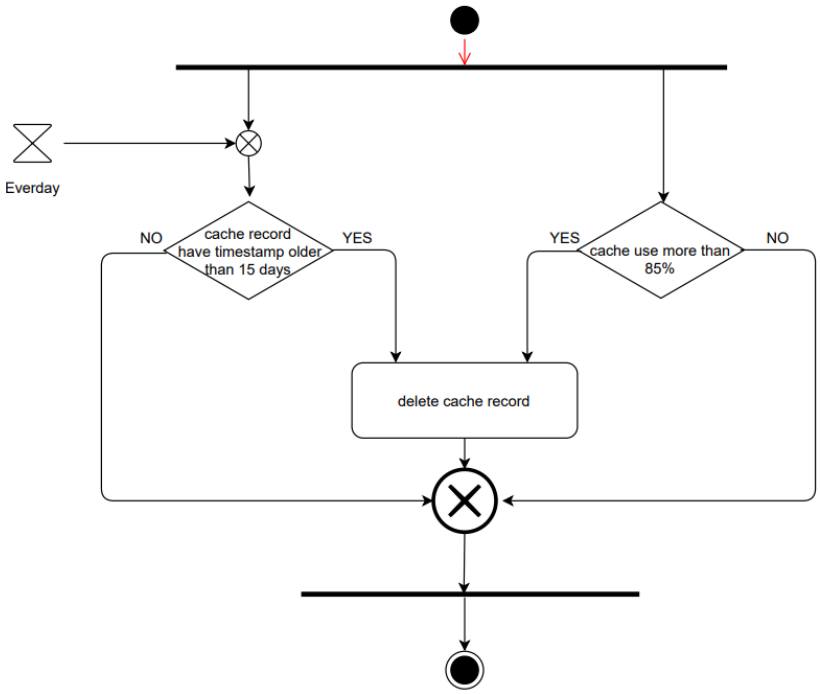
Call API End-Point



Reset Daily Limit



Cache Management



7) **PLANNING FOR FURTHER DEVELOPMENT**

We have generated issues on GitHub and assigned them to group members. As of now, we have formed sub-teams as per the below structure.

Backend: Manan, Shivam, Tanu, Paresh
Frontend: Keyur, Isha, Harshil, Shruti
Testing: Deven, Milan

PS: We will adjust the sub-teams in the future (if needed) based on the work.

8) **CONTRIBUTION**

We have done most of the tasks related to elicitation techniques and requirements gathering in the group. Each team member has collaboratively participated in every meeting.

Meetings done so far:

1. Shortlist Projects
2. Choose Top 3 Project Preferences
3. Analyzed our final picks thoroughly and written down points to make in the bidding process.
4. Explored the assigned project and discussed how we will proceed with the development of the system.
5. Work allocation and generated issues on GitHub.
6. Prepared Activity Diagrams for the system
7. Mid-Evaluation prep

Now, as we are moving to the development phase, we have assigned the work and every team member is working on them.

9) GitHub project repository

https://github.com/WebMetadataRetrieval/15-Web_Metadata_Retrieval