The project scope: We as a team will be implementing this project by taking a 360-degree look over it. We will be working on an API based interface which will be interacting with Github to fetch certain queries mainly Repository name, Contributors, Date of commit and many more. The last stage would be deployment of project. The technology we would be using will be AWS lambda as the need is serverless deployment.

Work Breakdown Structure (RASC) : We believe when everyone knows what exactly is expected of them, they are bound to be more accountable and sincere in their work.

R----- Responsible (Ms. Megha)

Responsible for leading and delivering the project and is accountable for the quality of decisions and delivered work. Will ensure everything is up to the point before to be sent for the approval.

Other Role: API management.

A----- Approval (Ms. Pooja Thukral)

Responsible of taking ultimate ownership of task delivery. Ensuring responsible resources are delivering the task as planned. Owner of task assignment, accountable for the inputs and approvals before the work is signed off.

Other role: AWS lambda (Deployment)

S----- Support (Ms. Rojalin, Mr. Anish)

Supporting in various concerns of the project. Would be working as the project collaborators who will be timely assisting us and also working actively in a support role.

Other roles:

Ms. Rojalin—AWS lambda, API

Mr.Anish---AWS lambda, API

C----- Consultant (Mr. Mohtadin khan, Mr. Akbar, Mr. Joel)

Subject matter experts who will be in constant touch for the smooth functioning of the project. Will be providing consultation and assistance on regular basis on every level—technical, administrative, approvals, inputs.

Other roles:

Mr. Mohtadin khan---API Design (Code)
Mr. Akbar---API implementation (Code)
Mr. Joel---AWS lambda(IAM, Amazon API gateway)

API is the messenger that delivers a request to the provider that you're requesting it from and then delivers the response back to you.
There are 3 main stages in API Lifecycle.
-->Design
-->Implementation
-->Management

-->Design stage includes:
Design=Identify process and business requirements, create logical data model, translate into logical service, API groupings
Simulate= Model API resources, model API operations/methods, model request/response payload/codes
Feedback= Mock up the API, publish interactive console, create notebook use cases, receive developer feedback
Validate= Modify API design as appropriate based on developer feedback, continue to validate

-->Implementation: Orchestration
Transformation
Routing
Data mapping
Connectivity across systems

-->Management stage includes:
Security

Deployment
Monitoring
Troubleshooting
Managing

Requirements:

Business requirements:

A document from the client stating the vision, mission, objective and what they wish the project to achieve so that a statement of need can be prepared for the project.

Functional Requirements:

Functional requirements define what the API does and how the API will be used.It is based upon the functionality given by the client(Example---To fetch a Gitlist we would be required a Source ID).

Non-functional Requirements:

Correctness-->The Ability with which the software respects the specification.
Performance-->The Ease with which the software is doing the work it is supposed to do. Usually measured as response-time or throughput.
Reliability-->Ability with which the software performs its required functions under stated conditions for a specified period of time.
Robustness-->Ability with which the software copes with errors during execution.
Security-->The Degree to which the software protects against threats.
Usability-->The Ease with which the software can be used by specific users to achieve specific goals.

Implementation Requirements: User Authentication, Data privacy, App Authentication.

AWS Lambda as a big compute resource manager:

When building applications on AWS Lambda the core components are Lambda functions and triggers. A trigger is the AWS service or application that invokes a function, and a Lambda function is the code and runtime that process events.

Lambda runs your code on highly available, fault-tolerant infrastructure spread across multiple Availability Zones (AZs) in a single Region, seamlessly deploying code, and **providing all the administration, maintenance, and patches of the infrastructure**.

AWS introduced the SAM (serverless architecture model) CLI. This is the tool for running serverless architectures, the core of which are Lambdas, locally.

ConstraintS :

Some constraints with Lambda

- AWS Lambda functions will timeout after 15 minutes which is fairly generous, but for level workloads, this might not be enough.
- More importantly the packaged functions are limited to 250mb unzipped and 50mb zipped which include the size of the packages, coded functions, and other dependencies.
- Ensure that the credentials you configured in include appropriate read/write access to the AWS Lambda service.

Some constraints with API:

- **Not found:** sent when the request was valid, but there's nothing available at the specified endpoint.
- **Server error:** sent when something is wrong with the API itself.
- **Authentication error:** sent when authentication is required but wasn't present in the request.
- **Invalid request:** sent when the user's request wasn't correctly formatted.
- One of the biggest challenges with API testing is the initial set-up. It's difficult to build an API and then have to go back to create tests after the API has been created. This becomes harder if the API design and build process were already strenuous.
- API users, can make requests with typos or mistakes that APIs just don't know how to handle.

Testing the API'S -----A successful run or not:

- API testing is a type of software testing that involves integration testing to determine if they meet expectations for functionality, reliability, performance, and security.
- The main drivers to invest in software testing are to ensure three main things: Functionality, Security, Performance.
- When testing APIs, test cases can be designed for each endpoint or a group of endpoints that are designed to work together (i.e a functional flow like logging in).