Repo-Schlep

Solution Design Document

REPO - SCHLEP

Overview

This Solution Design Document gives an insight on – executive summary, system overview, document organisation, design guidelines, design considerations and software architecture design of Repo-Schlep.

Introduction

Purpose

The purpose of Repo Schlep is to create an API which fetches the contributor statistics of any user specified Github ID. Those specified repositories will be sorted on the basis of commits for all branches.

Scope

The stakeholders will be able to provide an open source project to the users to sort the useful information and also fetch it for them. However, the end users will be able to save a lot of time and view the desired user/ repository.

Audience

This Solution Design Document prepared by our team, can be reviewed and revisited by our stakeholders (Deepak sir) at any point during the development of this project.

Executive summary

Goal

Our goal is to create an API gateway.

Generally API stands for Application programming interface which works as a messenger, as it conveys the information & responds back desired information to users.

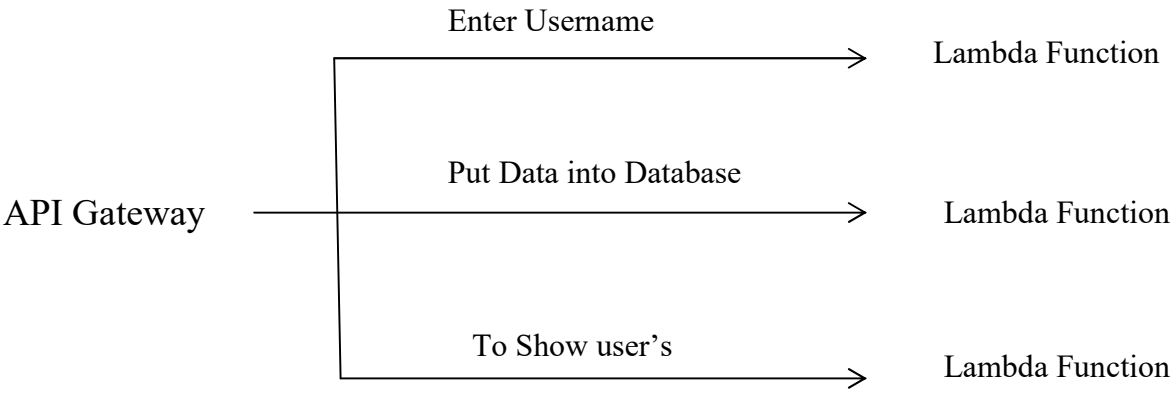
Objective

API technology is foundation of our project.

We are going to create API which will extract Github user data & put it into database & new webpage will appear where all repositories & details will be mentioned.

Strategy

We are using AWS platform. We will call API through lambda function. We are using relational database also. For database storage we will use SQL, it will store Github user's data



System overview

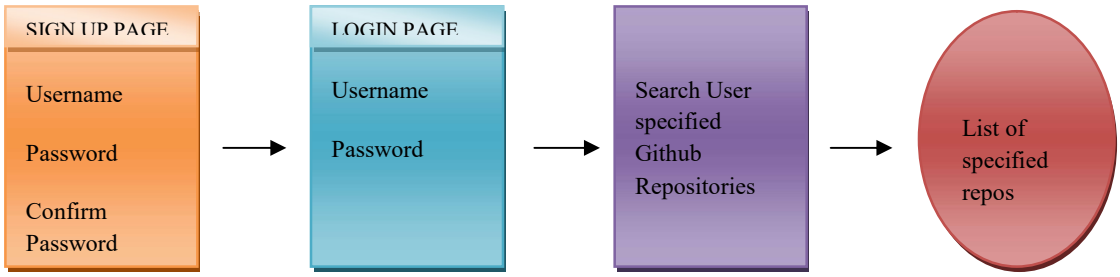
The Repo Schlep aims to fetch statistics of any Github contributor required by the user.

Key Features

- It allows the user to search for any contributors on Github.
- It sorts the contributors by number of commits.
- Includes user authentication by asking them to sign-in into their Repo Schlep account.
- Making it easier for our users to find the most relevant commits in the most efficient way is our motive.

Components

- Sign up Page – it will help the new users to sign up for Repo-Schlep.
- Login Page – it will help the existing users to authenticate themselves and use the services
- Search Page – it enables the user to



Document organisation

During this project, all the documentation will be updated on Github (https://github.com/apurvaapathak).

The repository called "Leap-4.0_MajorProject" contains all the required, necessary documentation related to the project. It includes the Business Requirement Document that was presented last week. We will be updating the repository as and when we keep preparing the documentation of this Repo-Schlep. It will include the Solution Design Document as well as the Customer Acceptance Document.

Apart from that, under the same Github link, we are also using a project called "Leap-4.0_MajorProject" which highlights the work flow of our team. It has different columns to represent – the tasks to be done, the tasks in progress, the completed tasks and the issues that we are facing.

Design guidelines

Roles and responsibilities

S. No.	Roles	Team Member(s) Responsible
1.	Documentation (Solution Design Document and Customer Acceptance Document)	Apurvaa Pathak Aakash Deep Yashovardhan Nasirabadkar Vedansh Dixit Vitthal Viksah Sinha
2.	Backend – using AWS and PHP	Monika Surana Priti Gupta Rakesh Gupta
3.	Designing – UI Design and Logo Design	Akashdeep Chand

System Assumptions

- The user will sign up to our website.
- The user will come back repeatedly to our website, login and generate traffic.
- Github does not face any glitches or technical difficulties.

Constraints

The constraints that can prevent the smooth functioning of Repo Schlep are, if we face more traffic than expected, if the Github server goes down due to any reason, it will directly affect our website.

Dependencies

Accumulation of dependencies may lead to slowing down or inefficiency of our API, therefore those are listed below:

- AWS – from creating to storing and also deployment, we are completely dependent on AWS servers to work efficiently.
- Github – to fetch the data that is required by our users, we also require Github to work and its server to not be down due to any technical difficulties.

Risk

Using Amazon Web Services as our core, the greatest risk can be the budget. If due to some overseen situation there is a misuse of any of the products or services of the said cloud service, we may be charged a huge amount of money.

Therefore, a careful supervision of the products and services being used is to be checked on a regular basis.

Design considerations

Operational Environment

- Throughout this project, our team will largely work on the Windows Operating System.
- We will use REST (Representational State Transfer) as our Web API which will make use of HTTP protocol.
- For storing the data, we will use AWS Relational Database Service.
- To connect the user to the database, i.e. to help him sign up or login, we are using PHP language.

Development Methods

To deliver a working and deployable product, we are using the AGILE development methodology. We will be consistently developing the product and testing different use cases simultaneously, in order to dynamically improvise the product. This methodology will help keep the project flexible as it will be easier to accommodate any changes required by the stakeholder.
We are also sharing the weekly progress with our documentation and will include the representation of a working prototype by the end of the next week.

Software architecture design

Our team has decided to use the Layered Software Architecture for this project. Through the Layered Architecture, we have attempted to provide specific functionality to each horizontal layer of the software. Two layers are being utilised on the backend – the database layer and the logic layer.
The frontend layer is called – the presentation layer