**Name:** Yash Panchamia **Topic:** Mini Project – Cross River Bank

1. I have tried and tested various Amazon Web Services Architecture while working with this project. They are:
2. AWS Architecture1:

AWS Lambda

(Event Driven)

API Gateway

(RESTful API)

AWS S3

(Front-end/UI))

A

AWS S3

(DataStore)

So, according to this architecture, what I tried to achieve is that when the user selects a year from the dropdown menu on the User Interface, that will hit lambda through API Gateway, Lambda will fetch the dataset from data store which is present in S3 and process the dataset and return the result to the User Interface via API Gateway. But, the problem with this architecture was that the dataset will be downloaded every time there is a user request of selecting a year from the UI. So, I thought about another way of doing this.

1. AWS Architecture2:

AWS Lambda

(Event Driven)

API Gateway

(RESTful API)

AWS S3

(Front-end/UI))

1. A

Dynamo DB

(Scalable NoSQL)

So, in this architecture, I made the change of replacing S3 with Dynamo DB. Dynamo DB a NoSQL database where I am storing my pre-processed dataset. So, whenever the user sends a request from the UI by selecting the year as input, Lambda will make fetches from Dynamo DB and return the results to the UI via API Gateway. Here, I went ahead and took the liberty of making a major assumption that our dataset (420 Mb file) is a static dataset. If the dataset is actually dynamic, then we can write an AWS service to communicate between Dynamo DB and S3 that whenever there is a change in the dataset, fire up an EC2 instance to pre-process the data and store it in the Dynamo DB. So, in this way, we will not lose the newly added entries either.

1. Goals Met:

* Use a scalable data store (e.g. AWS S3) to store the dataset.
* Develop a simple web service which processes the data from the store at runtime and returns the required aggregate loan parameters.
* Create a responsive web interface enabling the end user to select the year of interest from a drop-down list and a submit button
* Deploy the application to the public cloud platform to host the web-app and share the public IP address/domain (if available).

1. Goals Not Met:

* Leverage the web service to fetch the aggregate data and display a page of results similar to the given design (bonus for enhanced design).

Here, I have developed the UI using material design, designed the backend and integrated all the web services from the above architecture and integrated the backend with the UI. I also show TotaI Amount Applied for, Total Amount Funded and Total Committed by Investors cards where I fetch those values from Dynamo DB table ‘Loan’. I have hosted the web service on Amazon S3 and deployed the APIs from API Gateway. I couldn’t complete the data aggregation due to insufficient time and the conflicts and constraints in Lambda that when I tried to install pandas and numpy on Lambda, it always gave me errors because Lambda internally wants those packages to be of Linux form but what I was giving was of the Windows format. I also couldn’t complete the graphs on the UI.