

**An**  
**Industrial Training Report**  
**Submitted to**  
**UKA TARSADIA UNIVERSITY**  
**in partial fulfillment of the requirements for the degree of**  
**Bachelor of Technology Computer Engineering**  
**in**  
**By**  
**Harshita Kakadiya(201603100910017)**  
Under the Guidance of  
Ms. Purvi Tandel



**Department of Computer Engineering**  
**Chhotubhai Gopalbhai Patel Institute of Technology**  
**Uka Tarsadia University**  
**Bardoli - 394350**  
**June 2020**

Chhotubhai Gopalbhai Patel Institute of Technology  
Uka Tarsadia University  
Bardoli – 394350

## Company profile



Biz-Insights is an IT solutions company providing innovative, best-in-class consulting, IT solutions and Outsourcing services to local and global businesses. Biz-Insights brings together a deep understanding of the business and technology dimensions of an organization to harness information to improve decision-making, financial management, regulatory compliance and customer service.

We are a boutique digital transformation development company, offering a range of information technology services and offshore outsourcing services to clients such as Web Development, Mobile Application, UI/UX Designing, Data Warehousing, Data Mining, Data Analytics and other Data Science & Analytics services. End-to-end solutions are provided to our clients' business needs by combining business domain experience, technical expertise, top-grade knowledge of industry trends and the best methodology. This is achieved with the help of a sound team of skilled professionals who utilize their expertise and experience in helping the business reach new heights.

## **ACKNOWLEDGEMENT**

The internship opportunity we had with **Biz-Insights IT Solutions Surat** was a great chance for learning and professional development. Therefore, we consider ourselves very lucky as we were provided with an opportunity to be a part of it. We are also grateful for having a chance to meet so many wonderful people and professionals who led us through this internship period.

Bearing in mind previous we are using this opportunity to express our deepest gratitude and special thanks to the **Co-founder of the company Mr. Kunal Shah** who despite being extraordinarily busy with their duties, took time out to hear, guide and keep us on the correct path and allowing us to carry out the project at their esteemed organization and extending during the training.

We also extend our sincere appreciation to **Ms. Purvi Tandel** who provided his valuable suggestions and precious time in accomplishing our project report.

We perceive this opportunity as a big milestone in our career development. We will strive to use gained skills and knowledge in the best possible way and we will continue to work on their improvement, to attain desired career objectives.

The Director **Dr. R. V. Patil** and the Head of Computer Department **Dr. Devendra V. Thakor** and **all other faculty members** involved in making IDP as a part of the academic assessment to enable students to gain the aspiring work experience before graduating. Furthermore, distinguished appreciation also goes to the **Training and Placement Cell** of Uka Tarsadia University.

Lastly, we would like to thank the almighty and our parents for their moral support and our friends with whom we shared our day-to-day experience and received lots of suggestions that improved our quality of work.

**Harshita Kakadiya (201603100910017)**

## ABSTRACT

*During our industrial training at Biz Insights IT solutions, we have worked on different projects and various training activities by gaining knowledge with practical implementation. Overview of all the works as follows:*

**“Web Scraping with Python”** initiates with the scraping of product details from ecommerce websites Amazon and Flipkart for price comparison. After gaining knowledge of practical implementation next is the client project where we scrapped the address and contact details of sample data with 80+ car dealers of USA, then the most important and major project was the scraping of the reviews and ratings of 4200+ car dealers of different cities from the websites such as Google, Facebook, BBB.org, Cars.com, DealerRater, Yelp where we made python scraper for each of the websites and obtain the required results successfully.

**“COVID-19 Patient Prediction System with Machine Learning”** is the model that predicts if a person can have a probability or not of having COVID-19. With this, the person can self-assess and don't panic and be at home. It can also help prioritize the patients who really have coronavirus and help the health care workers who really have it and save kits for the needed and the system implementation includes the deployment of the machine learning model on Flask and further hosting the website on AWS EC2 instance.

**“AWS (Amazon Web Services)”** incorporates majorly with the five tasks where we worked with various AWS services such as AWS Lambda, AWS S3, AWS Kinesis, AWS Dynamo DB, AWS Cloud Trails, AWS Athena, API Gateway and practically implemented all the required services in each task.

Further in the training activities we have provided various prototype implementation for the major project modules such as “**Django Hotel Management System Modules**”, “**Flutter Quiz Application and Login Registration App**”, “**Web Scraping of Data and its Geocoding with the Awesome Table Visualizations**”, “**Daily Bible verse API Development in Node.js and Go language**” and extension with “**AWS RDS service**”, “**Python string processing mini-project** “ which is a prototype of the client system and “**Web RTC**” for video conferencing.

## TABLE OF CONTENTS

---

AcknoWledgEment.....	v
Abstract.....	vi
List of Figures.....	x
List of Tables .....	xiii
Chapter 1 INTRODUCTION TO internship areas .....	1
1.1 Overview of all the Learning Areas .....	1
1.2 Scope/Application of all the Learning Areas .....	2
Chapter 2 Training activities.....	4
2.1 Data Science and Analysis .....	4
2.2 Web Scraping with Python.....	5
2.3 AWS (Amazon Web Services) .....	6
2.3.1 Task:1 Customer Activity Tracker .....	6
2.3.2 Task:2 Customer Activity Tracker .....	7
2.3.3 Task:3 AWS CloudTrail .....	7
2.3.4 Task:4 Amazon Athena .....	8
2.3.5 Task:5 AWS Kinesis with Firebase.....	9
2.4 Django Framework of Python for Hotel Management System.....	10
2.5 Python string processing mini project.....	12
2.6 Daily Bible Verse API in Node.js and GO Lang .....	14
2.6.1 API with Node.js.....	14
2.6.2 API with GO Language.....	15
2.6.3 Daily Bible Verse API with AWS RDS.....	16
2.7 Flutter Framework for Mobile Application.....	17
2.8 Web scraping and Visualization with Awesome Table .....	18
2.9 WebRTC for video conference .....	20
2.10 Freelancer Website Development like Bark.com .....	21
2.10.1 UI Template Frontend Development of the website .....	21

2.10.2 Backend Development with MYSQL Database and API .....	24
Chapter 3 System Planning .....	26
3.1 Web Scraping with Python of 4200+ car dealers .....	26
3.1.1 System planning and software model for web scraping .....	26
3.1.2 Functional Requirements (Software Requirement Specification) .....	27
3.1.3 Modules of the system .....	27
3.1.3.1 Module-1:-Web Scraping with Python Beautiful Soap.....	27
3.1.3.2 Module-2:-Data Selection.....	27
3.1.3.3 Module-3:-Data Pre-processing and Tidying.....	28
3.1.3.4 Module-4:-Data Transformation .....	28
3.1.3.5 Module-5:-Data Mining .....	28
3.1.3.6 Module-6:-Interpretation.....	28
3.2 COVID-19 Patient Predictor System with Machine Learning.....	28
3.2.1 System planning and model for the machine learning.....	28
3.2.2 Functional Requirements (Software Requirement Specification) .....	30
3.2.3 Modules of the COVID-19 Patient Predictor System.....	31
3.2.3.1 Module-1:- Gathering of COVID-19 Symptoms .....	31
3.2.3.2 Module-2: - Model Training and Requirement.....	31
3.2.3.3 Module-3:- Webapp with Flask.....	32
3.2.3.4 Module-4:- Deploying the machine learning model to the flask .....	32
Chapter 4 System Design.....	33
4.1 Web Scraping with Python of 4200+ car dealers .....	33
4.1.1 System Process Flow Diagram .....	33
4.1.2 System Workflow Diagram.....	34
4.1.3 Activity Diagram.....	34
4.1.4 Data Flow Diagram (DFD) .....	35
4.1.5 Timeline for Address and Contact Scraping .....	36

4.1.6 Timeline for Reviews and Rating Scarping .....	37
4.2 COVID-19 Patient Predictor System with Machine Learning.....	37
4.2.1 Timeline Chart .....	37
4.2.2 Sequential Workflow Diagram.....	38
4.2.3 System Architecture Diagram .....	38
4.2.4 System Processing Workflow Diagram .....	39
4.3 AWS (Amazon Web Services) .....	40
4.3.1 Timeline Chart .....	40
Chapter 5 Implementation and testing .....	40
5.1 Web Scraping with Python of 4200+ car dealers .....	40
5.1.1 Software Requirements .....	40
5.1.2 Snapshots .....	41
5.2 Flutter Applications .....	46
5.2.1 Registration and Login Application with Flutter and Firebase .....	46
5.2.2 Quiz Application with Flutter .....	48
5.3 COVID-19 Patient Prediction System with Machine Learning .....	50
5.3.1 Module:1 Parameter selection and Final Input data.....	50
5.3.2 Module:2 Developing training model algorithm in Python .....	50
5.3.3 Flask Web app development .....	51
5.3.4 Deployment of application and testing .....	51
5.4 Django Hotel Management System Prototype for all Modules .....	53
5.4.1 Module-1 Registration Login with Twilio OTP.....	53
5.4.2 Module:2 Email Password Reset .....	54
5.4.3 Payment Gateway with stripe .....	55
5.4.4 ORM Postgre SQL Database .....	56
5.4.5 CRUD Operations for Content Management.....	57
5.5 AWS (Amazon Web Services) all task Implementation.....	59

5.5.1 Task-1:-Customer activity Tracker.....	59
5.5.2 Task-2: Customer Activity Tracker .....	61
5.5.3 Task-3 AWS Athena for Analysis.....	62
5.5.4 Task-4 AWS Cloud Trail with AWS Lambda and S3 Bucket .....	64
5.5.5 AWS Kinesis with Firebase .....	68
5.6 Test Cases .....	69
Chapter 6 Conclusion and Future Scope.....	72
Chapter 7 References .....	73

## LIST OF FIGURES

Figure 2.1 (a) Visualization with Tableau .....	5
Figure 2.1 (b) Jupyter Notebook testing and visualize .....	5
Figure 2.2 Architecture of web scraping e-commerce sites .....	6
Figure 2.3.1 Customer Activity Tracker workflow-1 .....	7
Figure 2.3.2 Customer Activity Tracker workflow-2 .....	7
Figure 2.3.3 AWS CloudTrail system flow .....	8
Figure 2.3.4 AWS Athena and Analysis workflow .....	9
Figure 2.3.5 AWS Kinesis with Firebase workflow .....	9
Figure 2.4 Road Map for Django HMS .....	11
Figure 2.5 (a) Python Script of string Processing .....	12
Figure 2.5 (b) Client Required Output .....	12
Figure 2.5 (c) Script and Output .....	13
Figure 2.6.1 (a) Daily Bible Verse MYSQL DB .....	14
Figure 2.6.1 (b) Node.js API on localhost .....	14
Figure 2.6.1 (c) Testing of Node.js API.....	15
Figure 2.6.2 (a) Go Lang API on localhost .....	16
Figure 2.6.2 (b) Testing of Go Lang API.....	16

Figure 2.6.3 (a) Daily Bible Verse AWS RDS DB.....	17
Figure 2.6.3 (b) Daily Bible Verse in AWS RDS DB .....	18
Figure 2.7 Flutter App development flow .....	18
Figure 2.8 (a) Awesome Table Visualization .....	19
Figure 2.8 (b) Awesome Table Visualization and CSV file .....	20
Figure 2.8 (c) Mapping Visualization .....	20
Figure 2.9 WebRTC video chat .....	20
Figure 2.10.1 (a) Homepage Display the Services .....	21
Figure 2.10.1 (b) Homepage Display the Locations .....	21
Figure 2.10.1 (c) Frequent Questions with respective Services.....	22
Figure 2.10.1 (d) Final Response message and Request Question to User.....	23
Figure 2.10.2 (a) API for various Services .....	24
Figure 2.10.2 (b) API for Loactions .....	24
Figure 2.10.2 (c) API for storing the User Response to MYSQL Database .....	25
Figure 2.10.2 (d) MYSQL Database stores the User Response .....	25
Figure 3.1.1 Crisp-DM model lifecycle .....	26
Figure 3.2.1 (a) Machine Learning Deployment Cycle .....	29
Figure 3.2.1 (b) Testing and Monitoring Cycle .....	30
Figure 4.1.1 Web Scraping with Python Process Flow Diagram .....	33
Figure 4.1.2 Web Scraping with Python Workflow Diagram .....	34
Figure 4.1.3 Web Scraping with Python Activity Diagram .....	35
Figure 4.1.4 (a) Web Scraping with Python DFD LEVEL 0 .....	36
Figure 4.1.4 (b) Web Scraping with Python DFD LEVEL 1.....	36
Figure 4.1.5 Web Scraping with Python Timeline chart-1 .....	36
Figure 4.1.6 Web Scraping with Python Timeline chart-2 .....	37
Figure 4.2.1 COVID-19 Patient Prediction System Timeline chart .....	37
Figure 4.2.2 COVID-19 Patient Prediction System Sequential Workflow Diagram .....	38
Figure 4.2.3 COVID-19 Patient Prediction System Architecture Diagram .....	38
Figure 4.2.4 COVID-19 Patient Prediction System Processing Flow Diagram .....	39
Figure 4.3.1 AWS Services all tasks Timeline chart .....	40
Figure 5.1.2 (a) Web Scraping with Python Input CSV data.....	41
Figure 5.1.2 (b) Scrapper for Google Ratings and Review .....	42
Figure 5.1.2 (c) Scrapper for bbb.org Ratings and Review .....	42
Figure 5.1.2 (d) Scrapper for cars.com Ratings and Review .....	42
Figure 5.1.2 (e) Scrapper for Dealer-rater Ratings and Review .....	43

Figure 5.1.2 (f) Scrapper for Facebook Ratings and Review .....	43
Figure 5.1.2 (g) Scrapper for YELP Ratings and Review .....	44
Figure 5.1.2 (h) Web Scraping with Python output and missing values script .....	45
Figure 5.2.1 (a) Registration Form Flutter .....	46
Figure 5.2.1 (b) Login Form Flutter.....	47
Figure 5.2.1 (c) Firebase Database Flutter .....	47
Figure 5.2.2 (a) Quiz App Home page Flutter .....	48
Figure 5.2.2 (b) Testing Quiz App Flutter .....	49
Figure 5.3.3 COVID-19 Patient Prediction System Flask app .....	51
Figure 5.3.4 (a) COVID-19 Patient Prediction System Testing .....	51
Figure 5.3.4 (b) COVID-19 Patient Prediction System Result display .....	52
Figure 5.4.1 Django Registration Login with OTP.....	53
Figure 5.4.2 (a) Django Email reset link.....	54
Figure 5.4.2 (b) Django Email password reset link .....	55
Figure 5.4.3 Django Stripe Payment Gateway.....	55
Figure 5.4.4 Django ORM Database .....	56
Figure 5.4.5 (a) Django Create Operation in HMS .....	57
Figure 5.4.5 (b) Django Delete Operation in HMS .....	58
Figure 5.4.5 (c) Django Update Operation in HMS .....	58
Figure 5.4.5 (d) Django Search Operation in HMS .....	58
Figure 5.5.1 (a) AWS Task-1 Flask app customer form .....	59
Figure 5.5.1 (b) AWS Task-1 Lambda Function .....	59
Figure 5.5.1 (c) AWS Task-1 API Gateway .....	60
Figure 5.5.1 (d) AWS Task-1 Output in S3 and Dynamo DB .....	61
Figure 5.5.2 (a) AWS Task-2 Input CSV file in S3 .....	61
Figure 5.5.2 (b) AWS Task-2 Lambda Trigger Event .....	62
Figure 5.5.3 (a) AWS Task-3 Input CSV file in S3 .....	62
Figure 5.5.3 (b) AWS Task-3 Input CSV file in S3.....	63
Figure 5.5.3 (c) AWS Task-3 Athena Query .....	63
Figure 5.5.3 (d) AWS Task-3 Athena Query Results .....	64
Figure 5.5.4 (a) AWS Task-4 Creating the Cloud Trails .....	64
Figure 5.5.4 (b) AWS Task-4 Viewing the Cloud Trails .....	65
Figure 5.5.4 (c) AWS Task-4 S3 Bucket the Cloud Trails .....	65
Figure 5.5.4 (d) AWS Task-4 Output in nested folders in S3 .....	67
Figure 5.5.4 (e) AWS Task-4 Output in parquet file in S3 .....	68

Figure 5.5.5 (a) AWS Task-5 Input data in Firebase .....	68
Figure 5.5.5 (b) AWS Task-5 Lambda Function .....	69
Figure 5.5.5 (c) AWS Task-5 Output CSV file in S3 .....	69

## **LIST OF TABLES**

Table 3.1.2 Software Requirements of web scraping with python .....	27
Table 3.2.1 Software Requirements of COVID-19 Patient Predictor System .....	31
Table 5.6 Test Cases .....	70
Table 5.6 Test Cases .....	71

# CHAPTER 1 INTRODUCTION TO INTERNSHIP AREAS

Various learning areas during the IDP internship as below:

- **Data Science and Analysis with Python** • **Web Scraping with Python** • **AWS Services**
- **Python mini project of string Processing** • **API with various language such as Node.js, Go lang** • **Django Python Framework** • **Machine Learning** • **Flutter Framework** • **Web Scraping with Awesome tables visualization** • **WebRTC for Video Conferencing**

## 1.1 Overview of all the Learning Areas

- **Data Science and Analytics with Python:** Data Science is a detailed study of the flow of information from the colossal amounts of data present in an organization's repository. It involves obtaining meaningful insights from raw and unstructured data which is processed through analytical, programming, and business skills. Data Analytics refers to the techniques to analyze data to enhance productivity and business gain. Data is extracted from various sources and is cleaned and categorized to analyze different behavioral patterns. The techniques and the tools used vary according to the organization or individual.
- **Web Scraping with Python:** Web Scraping is a technique employed to extract large amounts of data from websites whereby the data is extracted and saved to a local file in your computer or to a database in table (spreadsheet) format.
- **AWS (Amazon Web Services):** AWS is a well-established cloud platform that offers reliable, scalable, and inexpensive cloud computing services. AWS is already commercially used by some globally recognized organizations such as Netflix, Unilever, Samsung, MI, etc. Whether you are using Cloud to run applications that share photos to millions of mobile users or to support businesscritical operations, a cloud services platform provides rapid access to flexible and low cost IT resources.
- **Django Python Framework for Hotel Management System:** Django is an open-source python web framework used for rapid development, pragmatic, maintainable, clean design, and secures websites. A web application framework is a toolkit of all components need for application development.

- **API with various languages such as Node.js, Go Lang:** Node.js is a JavaScript runtime environment that runs the server-side. Within that environment, we can use JavaScript to build our software, our REST APIs and invoke external services through their APIs. Go language is a programming language initially developed at Google is a statically-typed language having syntax similar to that of C. It provides garbage collection, type safety, dynamic-typing capability, many advanced built-in types such as variable-length arrays and key-value maps. It also provides a rich standard library.
- **Machine Learning:** Machine learning is a tool for turning information into knowledge. In the past 50 years, there has been an explosion of data. This mass of data is useless unless we analyze it and find the patterns hidden within. Machine learning techniques are used to automatically find the valuable underlying patterns within complex data that we would otherwise struggle to discover. The hidden patterns and knowledge about a problem can be used to predict future events and perform all kinds of complex decision making.
- **Flutter Framework:** Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Windows, Mac, Linux, Google Fuchsia, and, the web. The first version of Flutter was known as codename "Sky" and ran on the Android operating system.

## 1.2 Scope/Application of all the Learning Areas

- Data Science is an umbrella term that covers several tools and technologies. It emerges in the application of systems such as internet search, gaming, recommendation systems, image, and data analytics is used to describe everything for online analytical processing.
- Web Scraping with Python describes our minor project where we have analyzed the data by scraping the product details between two E-commerce websites Amazon and Flipkart for the price comparison. Another was the client project where initially we have scrapped the address and contact details of the Car Dealers of the USA

provided the list in sample CSV files. One of the major and crucial main project tasks was the Scraping of the reviews and ratings of the 4200 car dealers of USA from the various website such as Google, Dealer Rater, BBB.org, cars.com, and Facebook.

- AWS (Amazon Web Services) we were assigned various tasks such as customer activity tracker where we have used the AWS services such as Lambda, S3, Athena for analysis, etc. Further, the AWS kinesis was used to read the data from firebase to kinesis and AWS Cloud Log Trails were generated which are to be stored on the S3 bucket for further analysis.
- With the Django Framework of Python we have given a road map of the Hotel Management system by developing the various prototype as per client requirements like Registration/Login with OTP, Payment Gateway with stripe, CRUD operation for content management, Schema for the ORM Database, EmailReset Password Feature.
- API with Node.js and Go Lang, as developed for the Bible verse Application as per the client requirement to get data using API in JSON from the appropriate MYSQL database and AWS RDS Database.
- In this COVID-19 pandemic while learning the basics in machine learning we came up with an idea to develop a system for patients' prediction as a victim of coronavirus to help them take necessary prevention measures based on the symptoms provided and developed the COVID-19 Patient Predictor System with machine learning.
- Flutter Framework is the Hybrid Mobile Application development tool that we are exploring and made the application such as Registration/ Login and Quiz App.
- Awesome Table Transform any data into beautiful, dynamic, and functional apps with our Awesome Table web app. This article is an introduction that presents how to simply and easily create interactive apps using data stored in Google Sheets and

Microsoft Excel 365. Awesome Table is a web application that displays data from a Google Spreadsheet into various types of nice views (cards, maps, table, Gantt view...). It's also possible to add different filters and styles to the view to customize it.

- Web Real-Time Communications (WebRTC) is a specification for a protocol implementation that enables web apps to transmit video, audio, and data streams between the client (typically a web browser) and server (usually a web server).

## CHAPTER 2 TRAINING ACTIVITIES

### 2.1 Data Science and Analysis

We have work with various tools available for Data Analytics as shown below with examples:

- Tableau is a visual analytics solution that allows people to explore and analyze data with simple drag and drop operations.

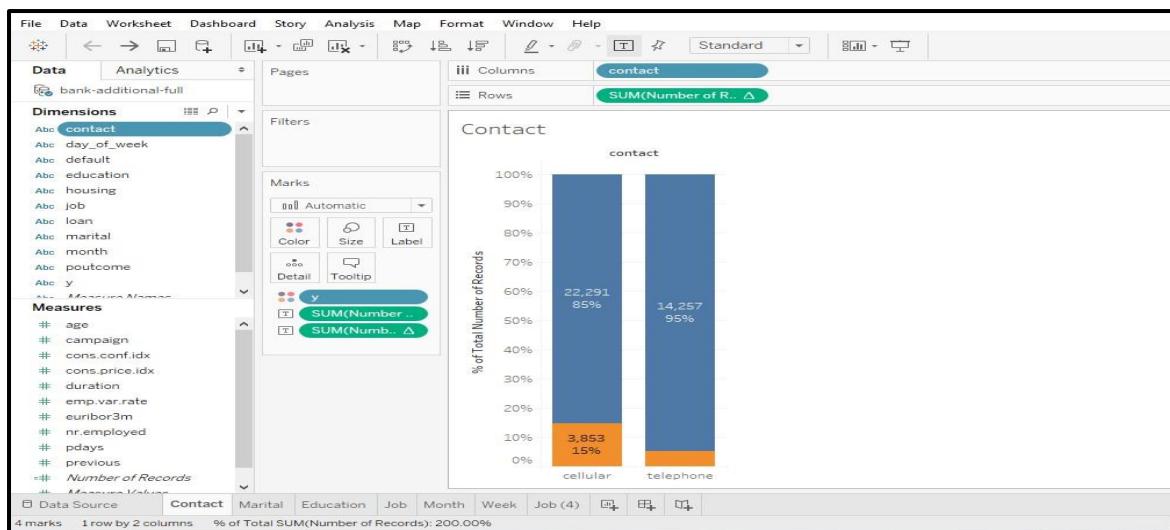


Figure 2.1 (a) Visualization with Tableau

- Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text.

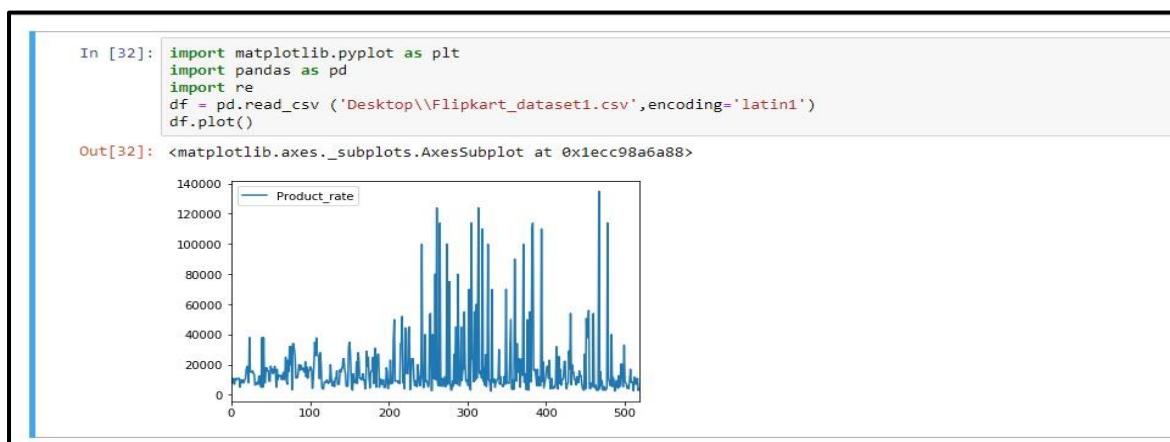


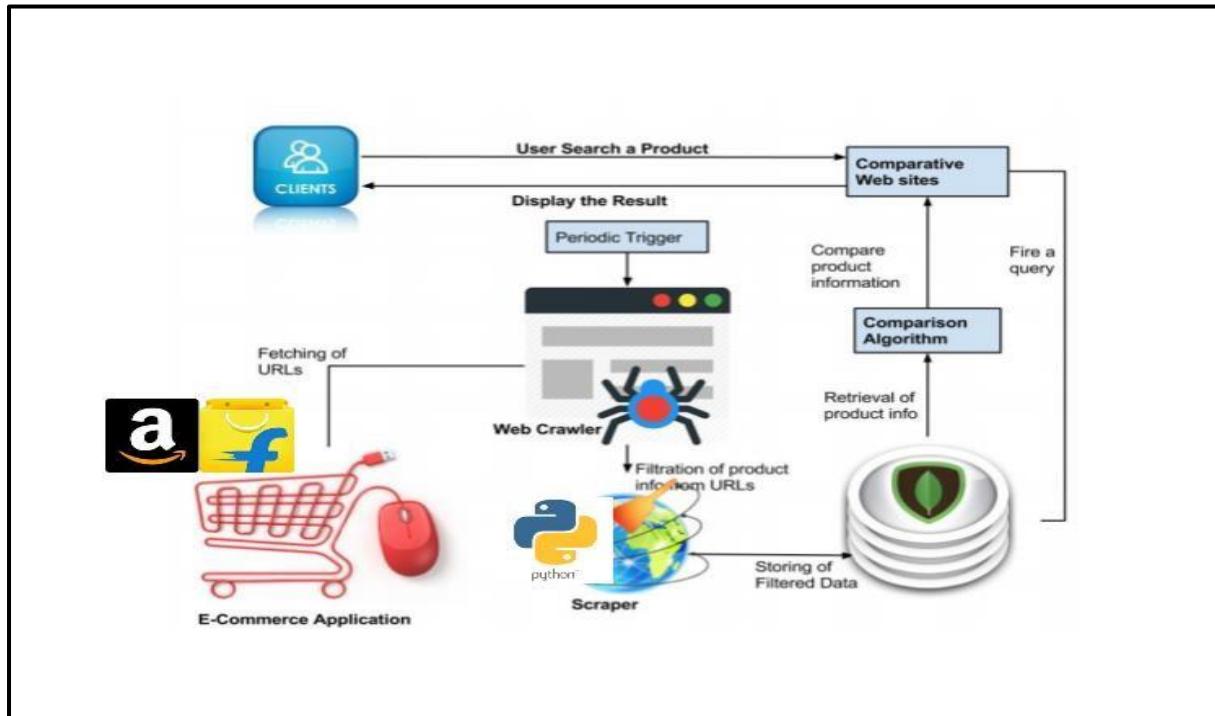
Figure 2.1 (b) Jupyter Notebook for testing and visualize

- Jupyter Notebook should be an integral part of any Python data scientist's toolbox. It's great for prototyping and sharing notebooks with visualizations and our working of further projects is with this tool and was very beneficial for learning for performing analysis in the professional world.

## 2.2 Web Scraping with Python

After gaining the theoretical knowledge of web scraping by following the various courses and video tutorials to experiment with the practical approach and implementation of web scraping with python we started working on the idea for scraping the product details from the **Amazon** and **Flipkart** we made the Amazon product scraper with python and Flipkart

product scraper with python and the product data scraped from each of site will be stored on the separate CSV file. Now as for the comparison the main parameter we selected was price comparison which will help all the customers to a larger extent with their favorite choice for buying products from both the websites. Also next we were assigned the Web scraping client project to scrape the reviews and ratings of around 4200+ car dealers of the USA will be discussed further.



**Figure 2.2 Architecture for web scraping with E-commerce websites**

## 2.3 AWS (Amazon Web Services)

### 2.3.1 Task:1 Customer Activity Tracker

Here we need to read data from the customer form and through AWS lambda which contains the core function act as API Gateway and stores the data in AWS S3 and the Dynamo DB.

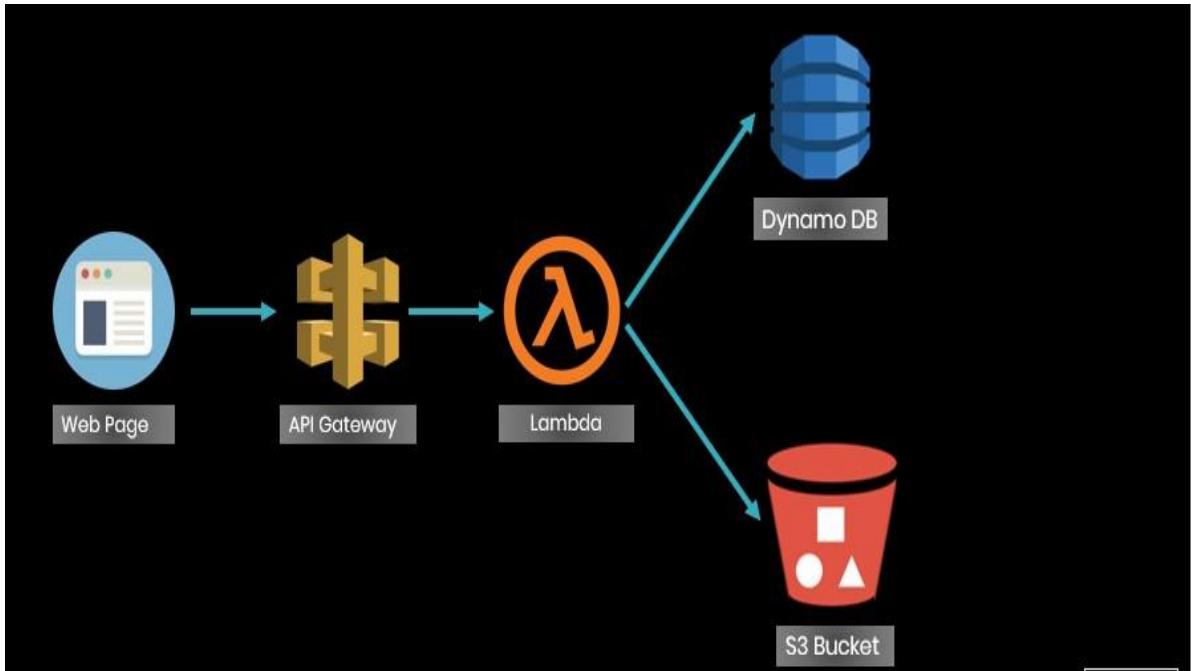


Figure 2.3.1 Customer Activity Tracker workflow-1

### 2.3.2 Task:2 Customer Activity Tracker

Here we need to read data from the CSV file stored in the S3 bucket and through AWS lambda which contains the core function act as API Gateway and stores the data in AWS S3 in compressed parquet format and the Dynamo DB.

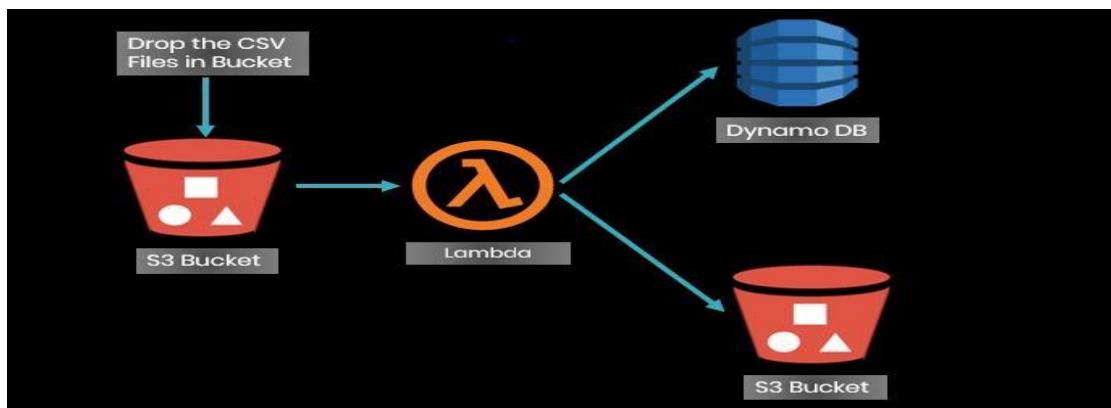
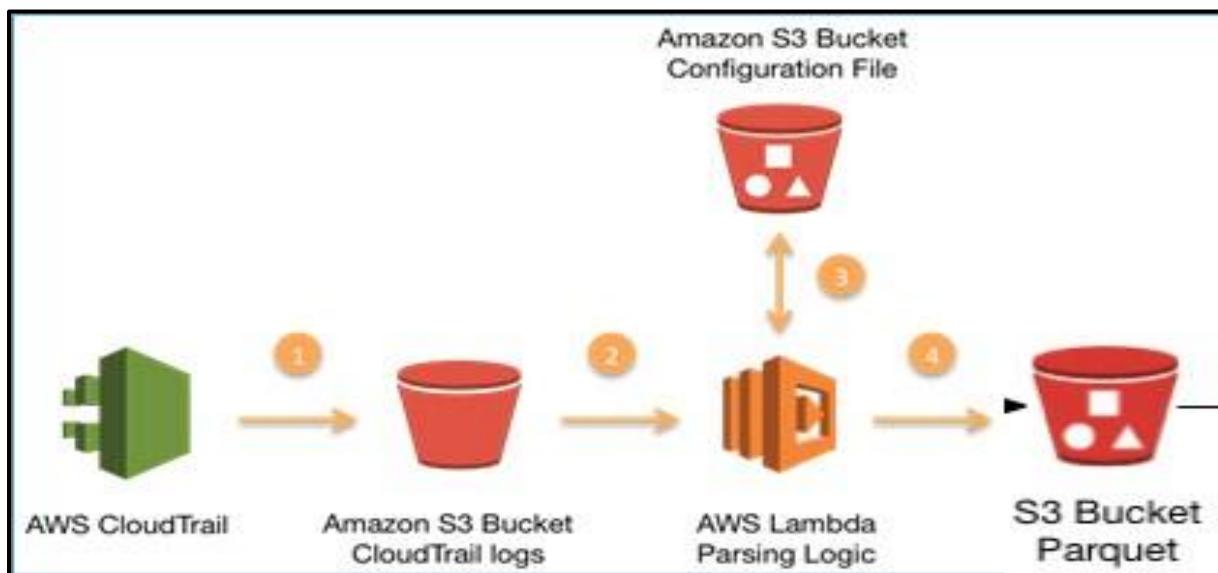


Figure 2.3.2 Customer Activity Tracker flow-2

### 2.3.3 Task:3 AWS CloudTrail

CloudTrail is a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures API calls as events. For an ongoing record of events in our AWS account, we create a trail. A trail enables CloudTrail to deliver log files of events to an Amazon S3 bucket and take advantage of Amazon S3's bucket notification feature and direct Amazon S3 to publish object-created events to AWS Lambda. Whenever CloudTrail

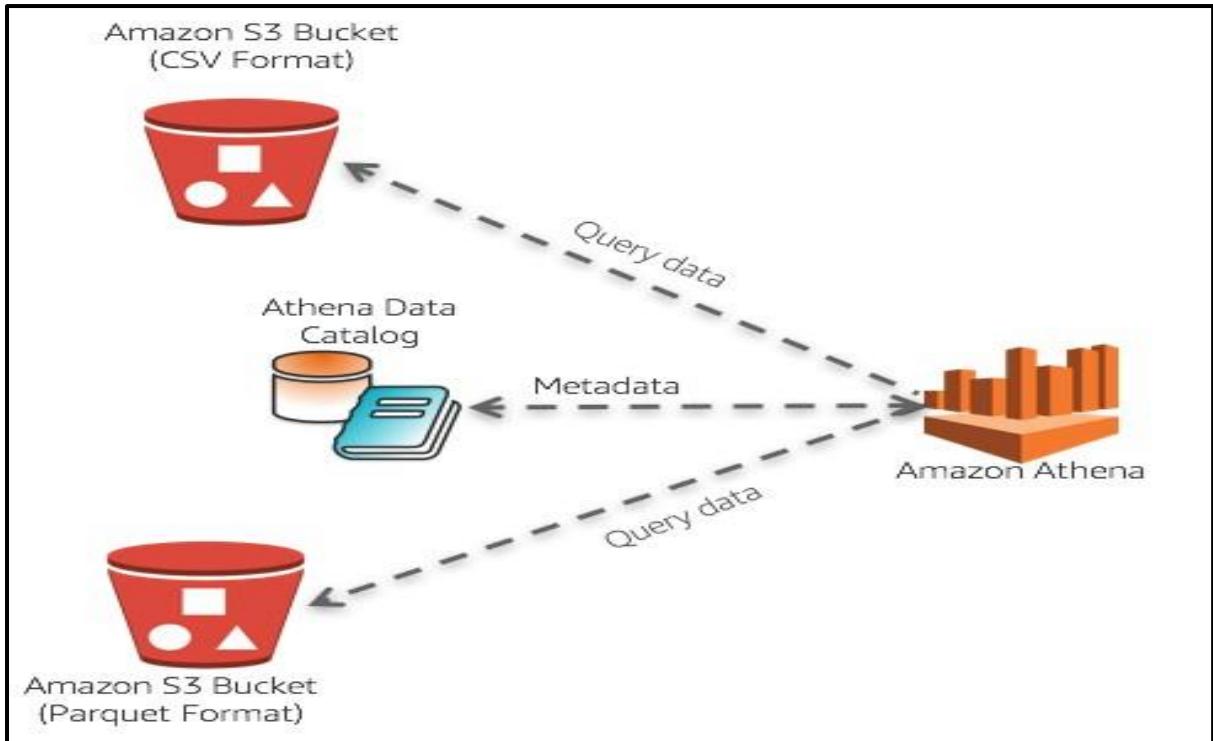
writes logs to our S3 bucket, Amazon S3 can then invoke our Lambda function by passing the Amazon S3 object-created event as a parameter. The S3 event provides information, including the bucket name and key name of the log object that CloudTrail created. Our Lambda function code can read the log object and process the access records logged by CloudTrail and finally stored the result in the S3 bucket in the form of a compressed parquet format.



**Figure 2.3.3 AWS CloudTrail system flow**

#### 2.3.4 Task:4 Amazon Athena

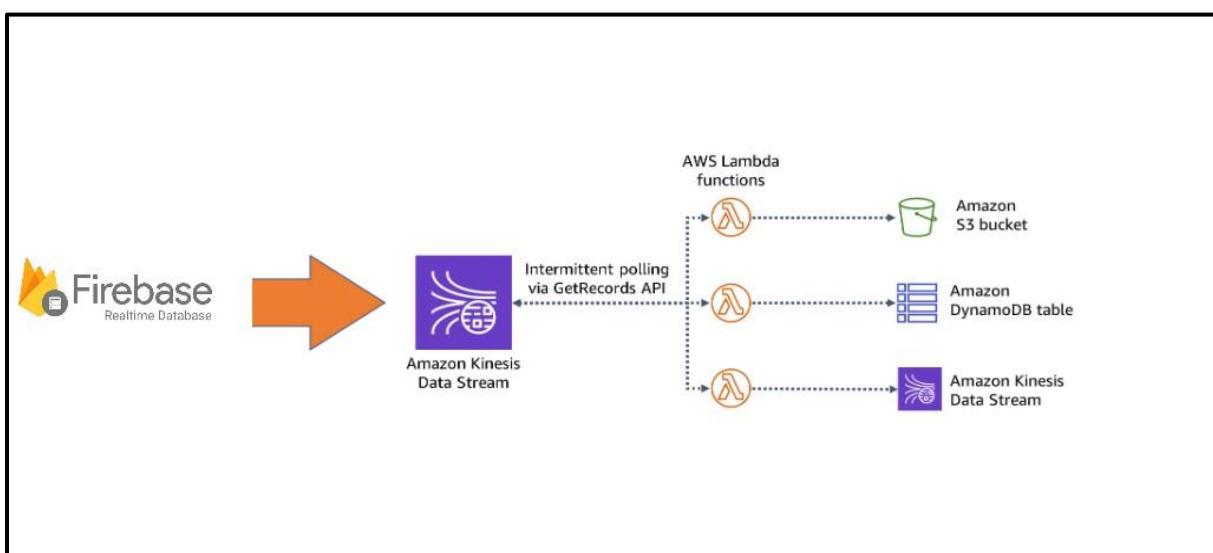
Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage. Athena is easy to use. Simply point to our data in Amazon S3, define the schema and we start querying using standard SQL. Most results are delivered within seconds and we analyze the various CSV files and parquet files for future use.



**Figure 2.3.4 AWS Athena Analysis flow**

### 2.3.5 Task:5 AWS Kinesis with Firebase

The task was quite challenging to complete as we need to merge the two cloud services from different platform one is the firebase from Google and AWS Kinesis from the AWS. The task is to read the data from the firebase real-time database and write the record to the AWS kinesis data stream and S3 using the AWS Lambda containing the core functionality that triggers to get the record and stores the data at the required destination.



**Figure 2.3.5 AWS Kinesis with Firebase flow**

## 2.4 Django Framework of Python for Hotel Management System

As per client project requirement, we were asked to give the road map and basic POC implementation of major modules from group discussions for the Hotel Management System to be developed with Django:-

**Module 1:- Registration and Login** is the basic module for implementation of any system hence we firstly develop the forms using crispy forms which is the package readily available in Django for the form design and further for secure and authentication we used the Twilio OTP service that generate the random OTP with the hash algorithm at backend and send the OTP notification to the user via, SMS.

**Module 2:- Email Reset Password feature** that helps the user when they forget their password. We have modified the app.py file and URLs .py file which is auto-generated in the Django project structure. Hence this feature helps the user at the time they forget the password by sending the Email password reset link so that they can again log in to the system after creating a new password for their account with recovering personal details.

**Module 3:- Payment Gateway with stripe** this module describes the secure payment method provided to the user. For the hotel management system, it plays a vital role in booking the hotels, reservation, and other services provided to the customers by various hotels so we have given prototype that user and pay securely by using stripe payment gateway.

**Module 4:- Database Schema for the Hotel Management System** includes the use of the Django ORM database which is more efficient than the traditional method of storing the user data. An ORM allows for simple manipulation of data without having to write complex SQL. In Django, the model is the object that is mapped to the database. When you create a model, Django executes SQL to create a corresponding table in the database without you having to write a single line of SQL. Django prefixes the table name with the name of the Django application for the Hotel Management System. After analyzing various databases we found that the Postgre database is compatible with Django in web development hence we used for the prototype of this module which stores all the user details and hotel details and easily provide help in managing the content.

**Module 4:- CRUD operations** this module plays a major role in any management system development hence considering it we provide the API that performs the CRUD operations such

as create, update, delete, search thus helps in the content management in the website. The content management helps the admin team to develop the system more efficiently for their users so this prototype plays a vital role in the Django Hotel Management System.

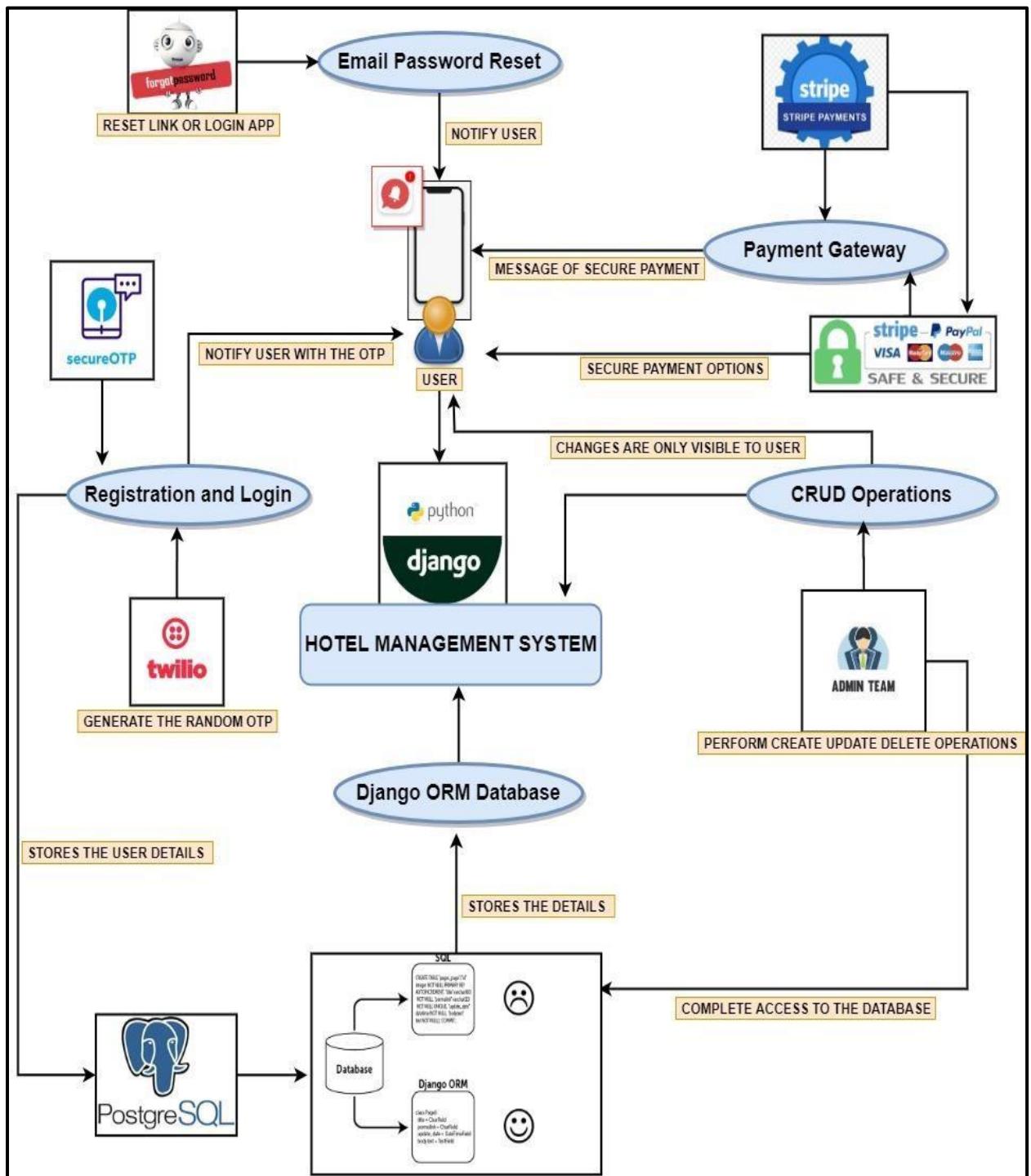


Figure 2.4 Road Map for Django Hotel Management System

## 2.5 Python string processing mini project

- Step:1 To process the input string using the Python script as it was initially developed the script in Nodejs takes a long time so the first step is for optimization.

```

import pandas as pd
strb = "sufyan"
# bad_chars = [';', ':', '!', "*", "."] 
# stra = "sufyan"
# test_string = filter(lambda i: i not in bad_chars, strb)
# stra = str(test_string)
# strb = "nayfus"
# stra = re.sub('[\W\_]', '', strb)
# stra = re.sub(r'^a-zA-Z0-9$', "", strb)
stra = re.sub(r"[?$_|_|!]", "", strb)
# stra = ''.join(e for e in strb if e.isalpha())
str_comp = ''.join(reversed(stra))
temp_str = str_comp
new_str = ""

new_str_list = []
positive_index = -1
negative_index = 0

list_of_string = []
list_of_string.append(str_comp)
counter = 1
switch_count = 0

while new_str != temp_str:
    for i in range(0, len(stra)):
        switch_count = switch_count + 1
        if(i%2 == 0):
            positive_index = positive_index + 1
            new_str_list.append(temp_str[positive_index])
        else:
            negative_index = negative_index - 1
            new_str_list.append(temp_str[negative_index])

    new_str = ''.join(reversed(new_str_list))

    if new_str == str_comp:
        list_of_string.append(new_str)
        break
    else:
        counter = counter + 1
        temp_str = new_str
        new_str_list.clear()
        positive_index = -1
        negative_index = 0
        list_of_string.append(new_str)
        new_str = ""

print("Total Rows: "+str(counter))
print("Total Count: "+str(switch_count))

```

Figure 2.5 (a) Python string process script

- Step:2 Python script developed for string processing further needs to be converted in the tabular structure to meet the Client requirements output below:

Type something...							
مؤخر المصدر				أول المصدر			
صغرى - 0 وسطى - 0 كبرى - 0 لفظي - 0 عدد الحروف - 36						نمبر شمار	
6	5	4	3	2	1	0	اساس
n	a	y	f	u	s	1	گردش
f	y	u	a	s	n	2	گردش
a	u	s	y	n	f	3	گردش
y	s	n	u	f	a	4	گردش
u	n	f	s	a	y	5	گردش
s	f	a	n	y	u	6	زمام
n	a	y	f	u	s	7	میزان

**Figure 2.5 (b) Client Required Output**

- **Step:3** Python script developed for string processing to for converting in the tabular structure using the data frame to meet the Client requirements and final result obtained as below:

```
#Converting to DataFrame
def split(word):
    return list(word)

splitted=[]
reverse_column=[]
last_column=[]

#for splitting of Strings
for i in list_of_string:
    splitted.append(split(i))

#for column
for i in range(0,len(list_of_string)):
    reverse_column.append(i)

#for last column
for i in range(1,len(list_of_string)+1):
    last_column.append(i)

df_last_col=pd.DataFrame(last_column)
reverse_column.sort(reverse=True)
reverse_column.pop()

df = pd.DataFrame(splitted,columns=reverse_column)
df['0']=df_last_col

df_without_index=df.to_string(index=False)
```

```
Total Rows: 6
Total Count: 36

Out[3]:
  6 5 4 3 2 1 0
  0 n a y f u s 1
  1 f y u a s n 2
  2 a u s y n f 3
  3 y s n u f a 4
  4 u n f s a y 5
  5 s f a n y u 6
  6 n a y f u s 7
```

**Figure 2.5 (c) Python script and Output**

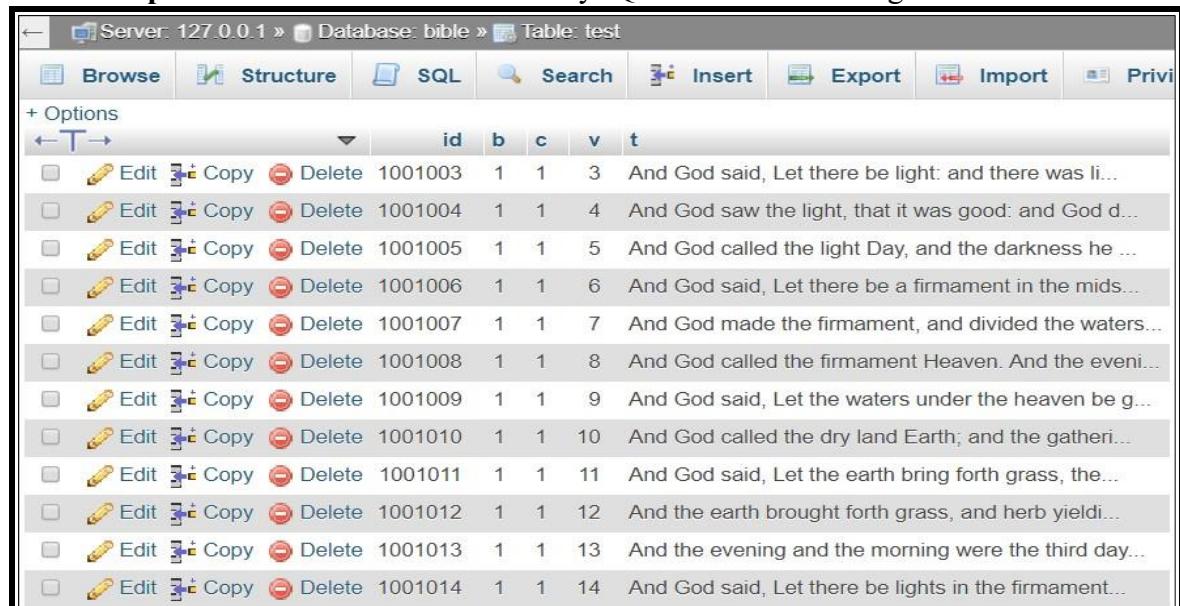
## 2.6 Daily Bible Verse API in Node.js and GO Lang

We have all the bible verses in our MySQL database and our Bible API gives an easy way to access the data for the Daily Verse Bible application.

**(Note: The common database used is MySQL to store and retrieve the data for both languages API)**

### 2.6.1 API with Node.js

- **Step 1:** - To create the database in MySQL as shown in the figure below:



			<b>id</b>	<b>b</b>	<b>c</b>	<b>v</b>	<b>t</b>
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	1001003	1	1	3 And God said, Let there be light: and there was li...
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	1001004	1	1	4 And God saw the light, that it was good: and God d...
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	1001005	1	1	5 And God called the light Day, and the darkness he ...
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	1001006	1	1	6 And God said, Let there be a firmament in the mids...
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	1001007	1	1	7 And God made the firmament, and divided the waters...
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	1001008	1	1	8 And God called the firmament Heaven. And the eveni...
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	1001009	1	1	9 And God said, Let the waters under the heaven be g...
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	1001010	1	1	10 And God called the dry land Earth; and the gatheri...
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	1001011	1	1	11 And God said, Let the earth bring forth grass, the...
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	1001012	1	1	12 And the earth brought forth grass, and herb yieldi...
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	1001013	1	1	13 And the evening and the morning were the third day...
<input type="checkbox"/>	<a href="#">Edit</a>	<a href="#">Copy</a>	<a href="#">Delete</a>	1001014	1	1	14 And God said, Let there be lights in the firmament...

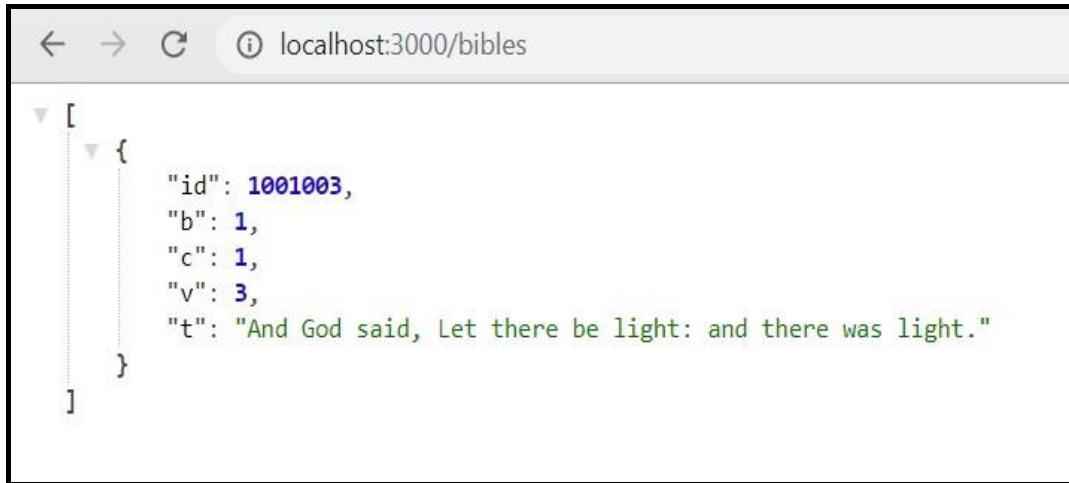
Figure 2.6.1 (a) MYSQL Database

- **Step 2:** - Building the Bible verse API in node.js to retrieve records from the database bible created above.

```
C:\Users\npate\Documents\idp\Nodejs>nodemon index.js
[nodemon] 2.0.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Express server is running at port number: 3000
DB connection successful
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Express server is running at port number: 3000
DB connection successful
```

Figure 2.6.1 (b) Node.js API on localhost

- **Step 3:-** Testing the API which successfully fetches the verses randomly on localhost port 3000.



```
[{"id": 1001003, "b": 1, "c": 1, "v": 3, "t": "And God said, Let there be light: and there was light."}]
```

**Figure 2.6.1 (c) Output with Node.js API**

### 2.6.2 API with GO Language

- **Step 1:-** Refer to the database in MySQL as shown in the figure above.
- **Step 2:-** Building the Bible verse API in Go language to retrieve records from the database bible.

```
C:\Projects\GO\src\sqlconnect>go run main.go
2020/03/24 11:38:23 {1002012 1 2 12 And the gold of that land is good; there is bdellium and the onyx stone.}
2020/03/24 11:38:37 {1003013 1 3 13 And the LORD God said unto the woman, What is this that thou hast done? And the woman said, The serpent beguiled me, and I did eat.}
2020/03/24 11:38:52 {1001005 1 1 5 And God called the light Day, and the darkness he called Night. And the evening and the morning were the first day.}
```

**Figure 2.6.2 (a) API with GO Lang on localhost**

- **Step 3: -** Testing the Go API in Postman which successfully fetches the record randomly as shown in the snapshot below and gives the required result.

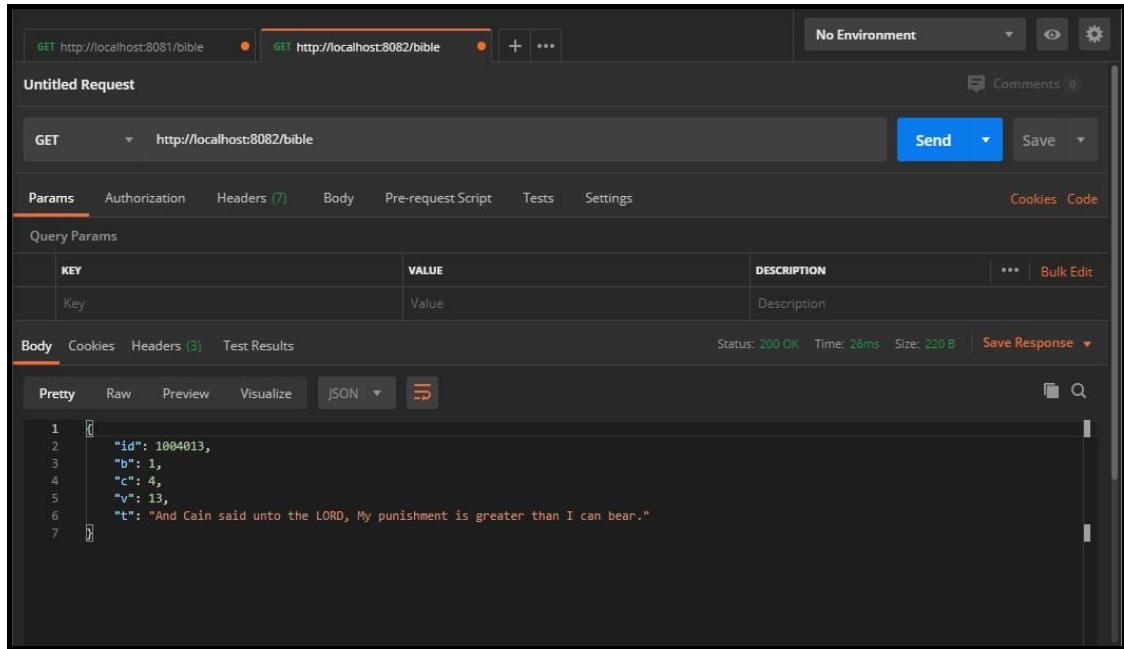


Figure 2.6.2 (b) Testing API with GO Lang

### 2.6.3 Daily Bible Verse API with AWS RDS

The bible verse API developed using both the above language and after testing was successful we have stored on the cloud service AWS RDS so the data are easily accessible to any user of the application using API with the cloud database.

The screenshot shows the AWS RDS console for the database1 instance. The instance is running MySQL Community engine, db.t2.micro class, and us-east-1c region. It has 2 connections. The security group is default (sg-1fa34636) and is active. The endpoint is database1.cnepbyvm40nk.us-east-1.rds.amazonaws.com.

DB identifier	CPU	Info	Class
database1	1.97%	Available	db.t2.micro

Role	Current activity	Engine	Region & AZ
Instance	2 Connections	MySQL Community	us-east-1c

Figure 2.6.3 (a) AWS RDS Database

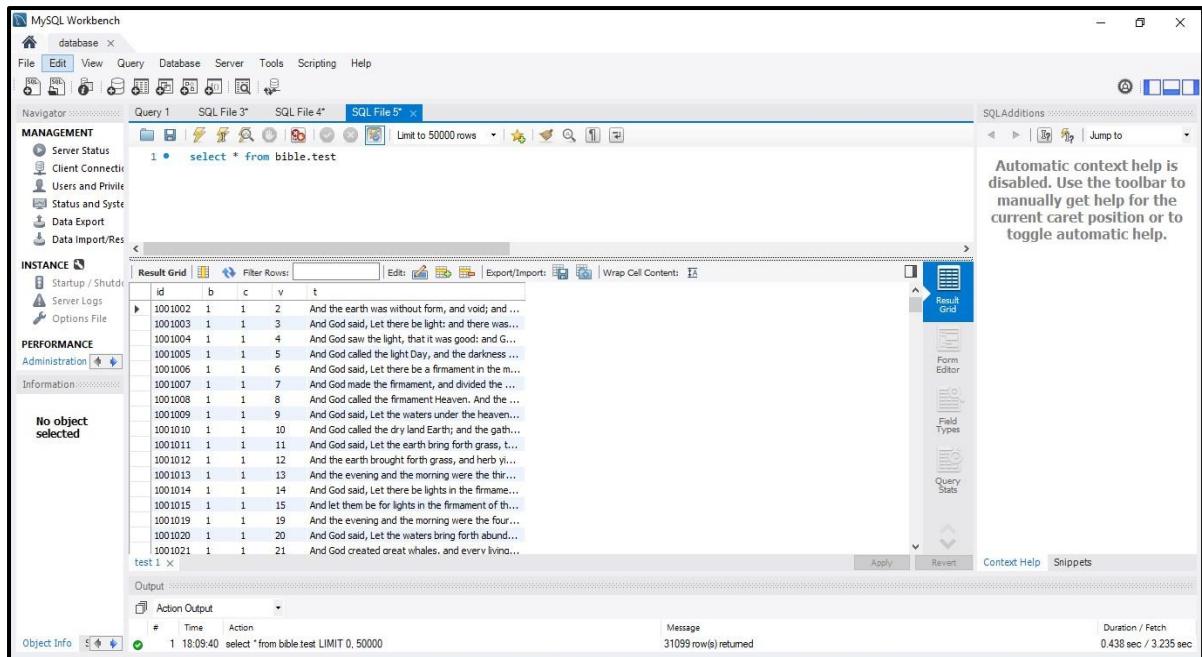


Figure 2.6.3 (b) Daily Bible Verse in RDS Database

## 2.7 Flutter Framework for Mobile Application

After exploring and gaining knowledge of the Flutter framework for the mobile application we have designed and implemented the two applications where the flow can be shown as below:

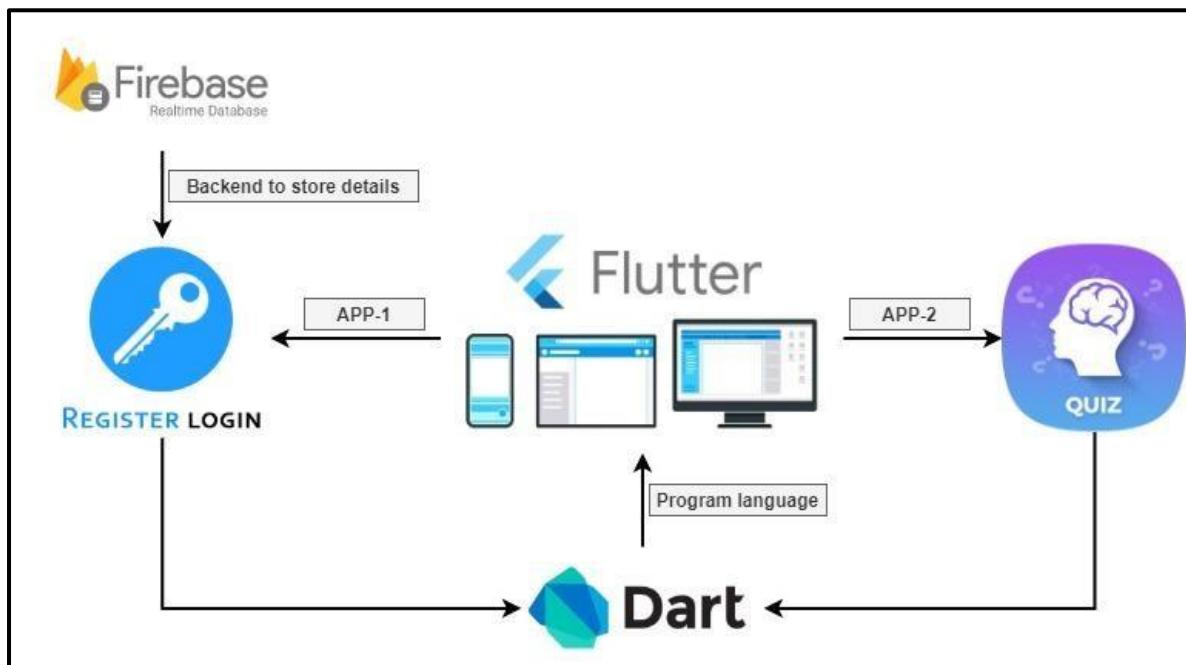


Figure 2.7 Flutter app development flow

## 2.8 Web scraping and Visualization with Awesome Table

Any data can be transformed into beautiful, dynamic, and functional apps with the Awesome Table web app. We have installed the add-ons directly within the Google Sheets user interface. Add-ons answer to specific needs like geocoding addresses, listing files or folders from Google Drive, and listing Google Sites on a G Suite domain. Hence using this interesting feature we get the following result for the patient's list of COVID19 of Surat.

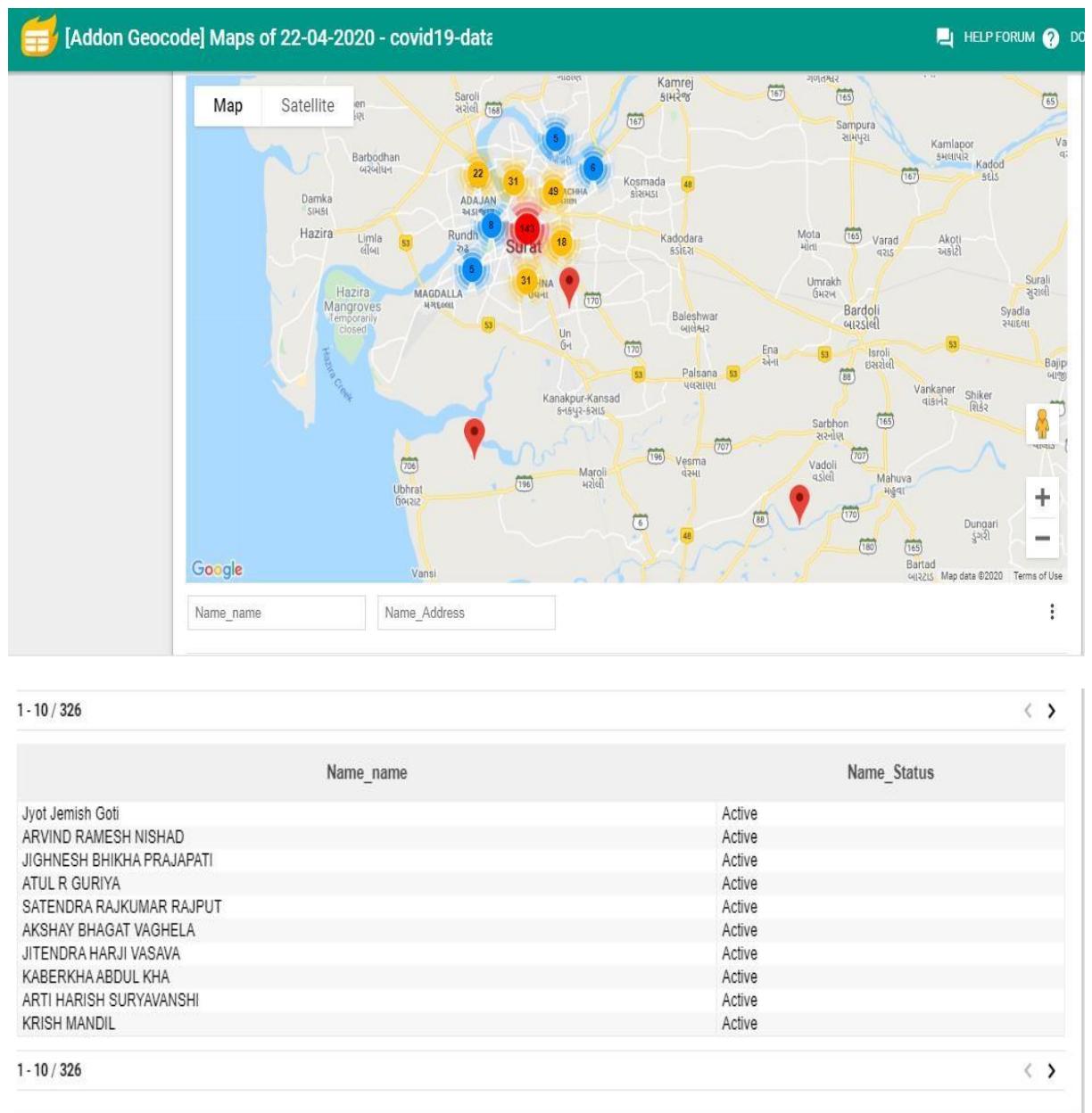


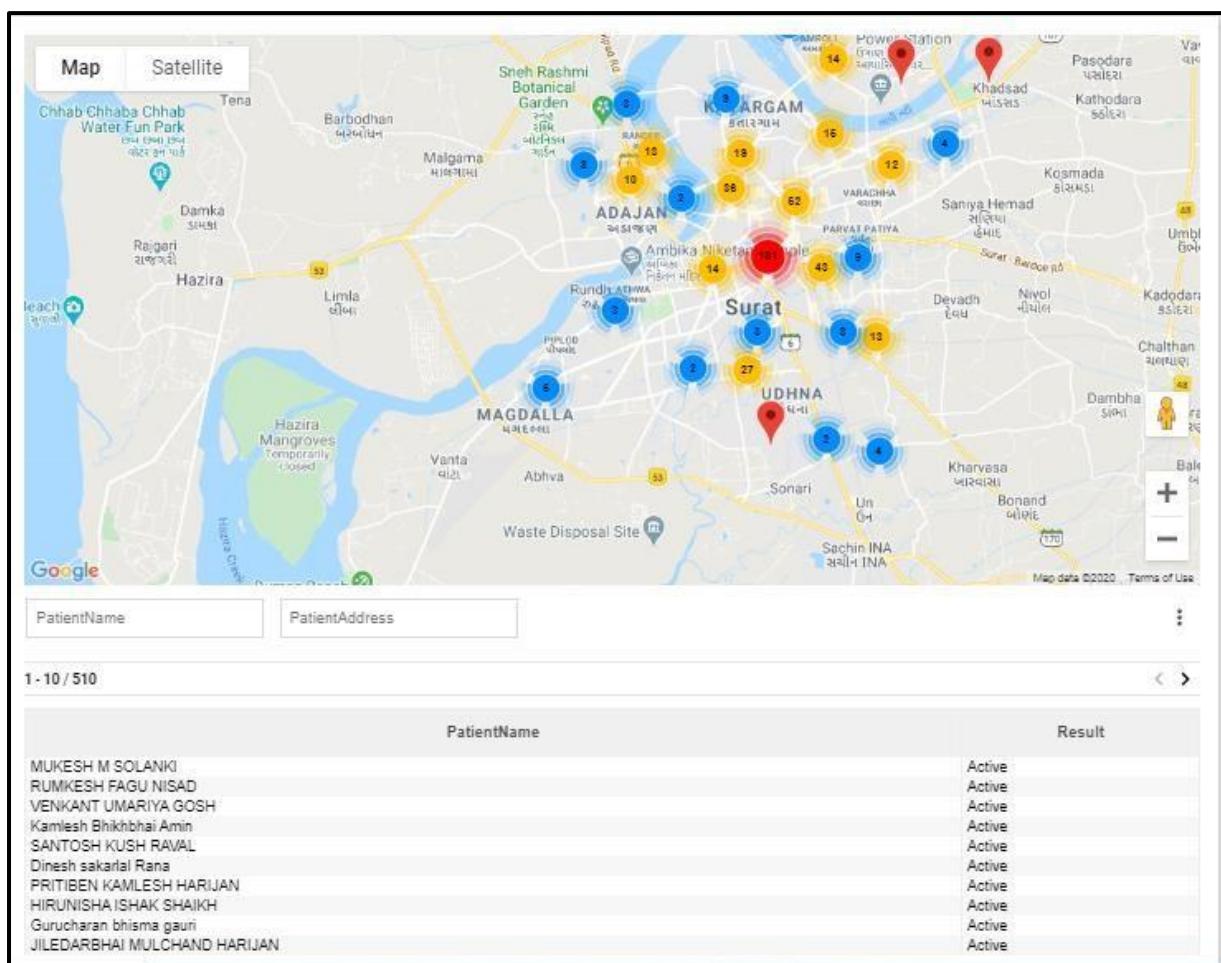
Figure 2.8 (a) Awesome table Visualization

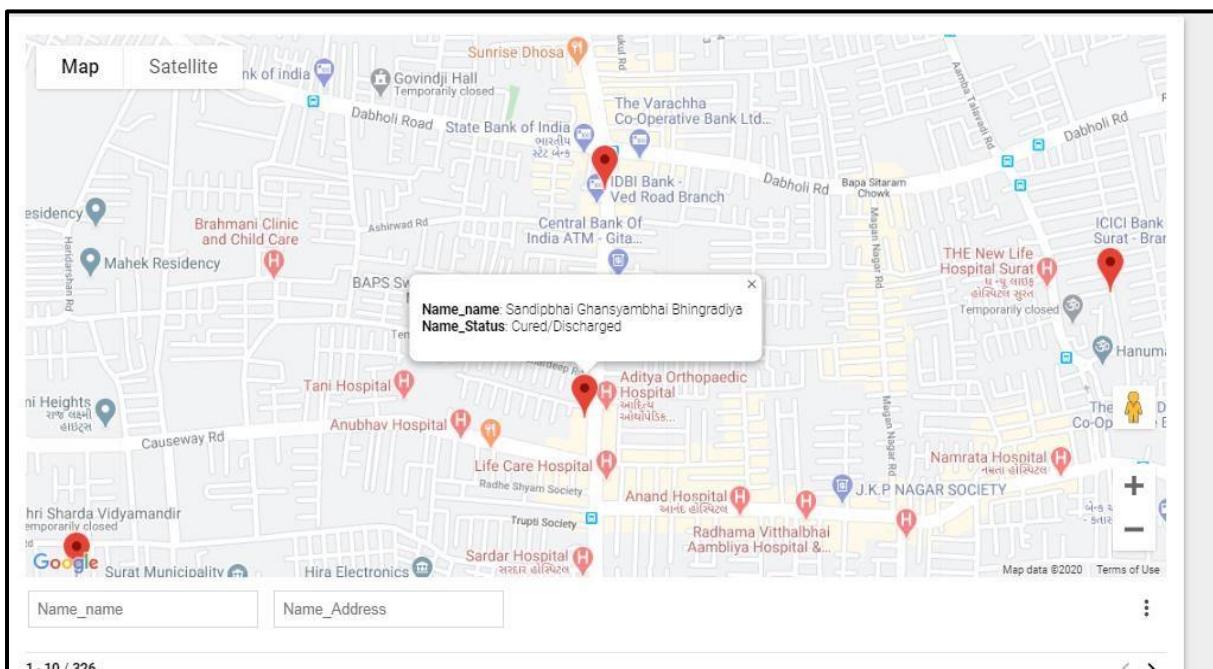
# Industrial Training Report

File Home Insert Page Layout Formulas Data Review View Developer Help Tell me what you want to do

A1 PatientName Result Latitude Longitude PatientAddress E F G H I J

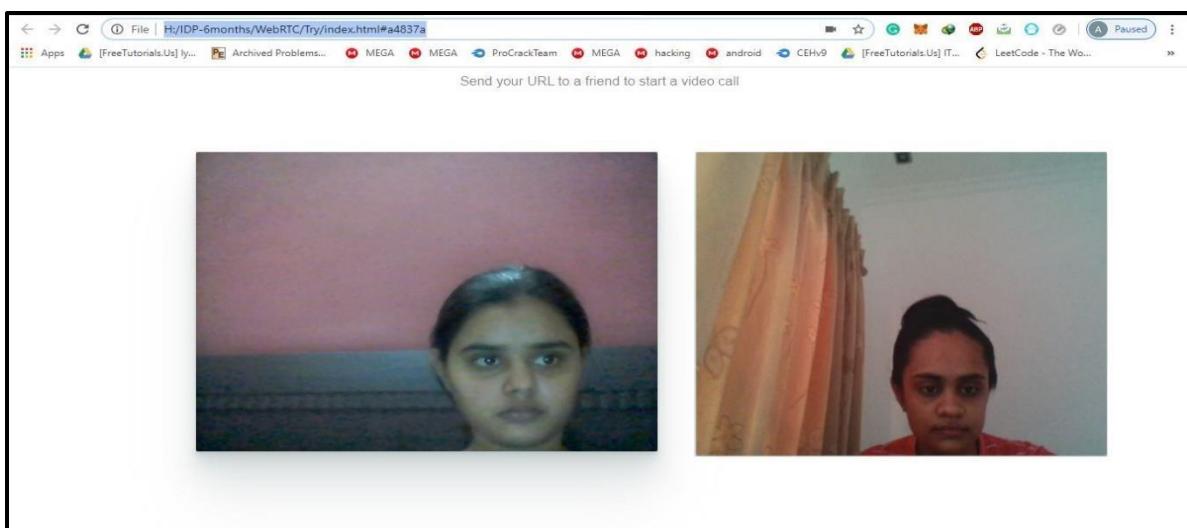
PatientName	Result	Latitude	Longitude	PatientAddress	ResultDate	city
MUKESH M SOLANKI	Active	21.24104	72.8587	70,71 HARI DARSHAN SOC, AMROLI	26-04-2020	Surat
RUMKESH FAGU NISAD	Active	21.21579	72.81514	RAMESHWARI SOC, VED ROAD	26-04-2020	Surat
VENKANT UMARIYA GOSH	Active	21.18226	72.8493	436, AMBEDKAR NAGAR, BHATHENA, SURAT	26-04-2020	Surat
Kamlesh Bhikhbhai Amin	Active	21.21639	72.82889	Prajapati samaj ni wadi,katargam	26-04-2020	Surat
SANTOSH KUSH RAVAL	Active	21.2188	72.8527	423 ATALJI NAGAR 412, Vallabhbacharya Rd, Atalji Nagar, Surya Nagar Society, Anand Na	25-04-2020	Surat
Dinesh sakarlal Rana	Active	21.19209	72.83428	847 dhamlawad sarabatpura	25-04-2020	Surat
PRITIBEN KAMLESH HARIJAN	Active	21.21922	72.8542	110 ATALJI NAGAR A.K ROAD 283, Kashama Society, Anand Nagar, Surat, Gujarat 395006	25-04-2020	Surat
HIRUNISHA ISHAK SHAIKH	Active	21.1806	72.8551	734,gali no 11,azad nagar ,limbayat	25-04-2020	Surat
Gurucharan bhimsa gauri	Active	21.17898	72.86651	27, krishnakripa society, Parvat gam	25-04-2020	Surat
JILEDARBHAI MULCHAND HARIJAN	Active	21.21923	72.85427	110 ATALJI NAGAR A.K ROAD 283, Kashama Society, Anand Nagar, Surat, Gujarat 39500	25-04-2020	Surat
DIPAK ANANTRAY DABHARE	Active	21.2432	72.8507	57 BHAGUNAGER 1, SAYAN ROAD AMROLI 13, Bhagu Nagar-1, Amrli, Surat, Gujarat 39	25-04-2020	Surat
Padmaben bhavshankar marathi	Active	21.1836	72.8358	2/13, B.Tenement mandaravaja, Umarvada	25-04-2020	Surat
ZINA KALIDAS JADAV	Active	21.21556	72.82373	105,GOKULDHAM SOC,VED ROAD	25-04-2020	Surat
DHAVALD PARMAR	Active	21.18042	72.83512	243/11, SHASTRI NAGAR, KHTODARA, UDHNA SURAT	25-04-2020	Surat
CHIRAGBHAI.D.BHATT	Active	21.25095	72.85875	B/8-301,SWEET HOUSE,RAJWADI PARTY PLOT,AMROLI	25-04-2020	Surat
KASTURIBEN MADHU DIVARE	Active	21.18242	72.83671	766 GALI NO-15V PADMANGAR,MANDARWAJA	25-04-2020	Surat
RANJANA Rajendra RAI	Active	21.20954	72.87629	P.N. 9, SHRENATHJI SOC., PUNAGAM, SURAT	25-04-2020	Surat
DR.DHARMESH KAUSHIK PATEL	Active	21.20135	72.79896		25-04-2020	Surat
CHANDKHAN SATTARKHAN PATHAN	Active	21.18116	72.8382	148 PANCHSIL NAGAR-1 BHATHENA SURAT	25-04-2020	Surat
DAXABEN KETAN GAYWALA	Active	21.17632	72.84576	NO 12, L.M.PARK, near Tara vidhyamandir school, BHATHENA, SURAT	25-04-2020	Surat
LALITA NILKANTH RATHOD	Active	21.1562	72.8321	P.N. 98, SHIVAJI NAGAR, PANDESARA	25-04-2020	Surat
FENIBEN DALAL	Active	21.15443	72.76565	22,UMA ROW HOUSE OPP PARSURAM GARDEN ADAJAN SURAT	25-04-2020	Surat
SAMABI SHEIKH	Death	21.18189	72.85298	GALI 2/349, 108 300, AZAD CHOWK, LIMBAYAT	25-04-2020	Surat
YOGITA S AHIRE	Active	21.20954	72.87628	P.N. 9, SHRENATHJI SOC., PUNAGAM, SURAT	25-04-2020	Surat
SHANKAR MANsingh	Active	21.22042	72.85566	SATYAM SHIVAM SUNDARAM APT, AK ROAD	25-04-2020	Surat
Sanjuben Maganbai Dobariya	Active	21.23215	72.8189	11 manishnagar 1 dabholi char rasta	25-04-2020	Surat



**Figure 2.8 (b) Awesome table visualization and CSV file****Figure 2.8 (c) Mapping Visualization**

## 2.9 WebRTC for video conference

As coronavirus lockdowns have moved many in-person activities online, the use of videoconferencing solutions is being increased day by day and one solution to videoconferencing is WebRTC. With WebRTC, you can add real-time communication capabilities to your application that works on top of an open standard. It supports video, voice and generic data to be sent between peers, allowing developers to build powerful voice- and video communication solutions. The technology is available on all modern browsers as well as on native clients for all major platforms.



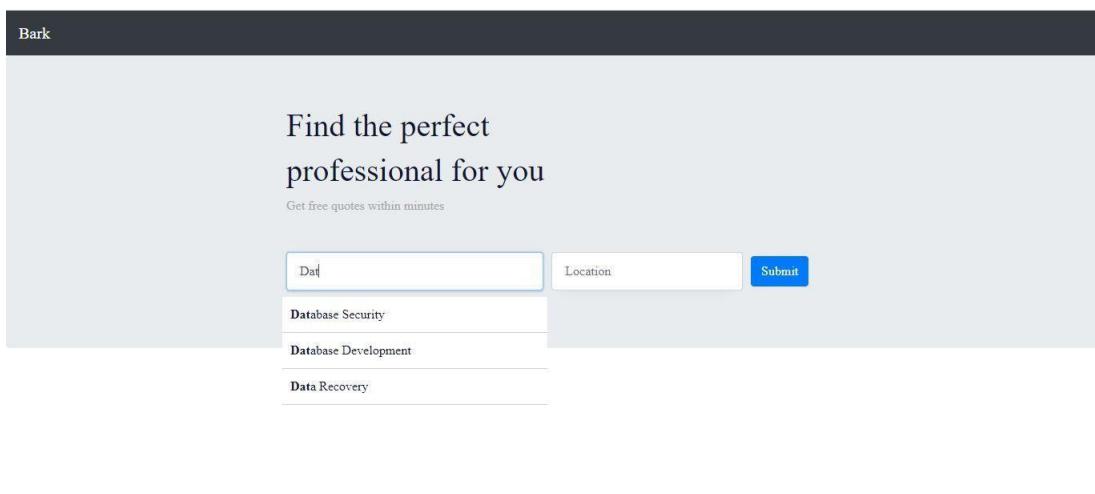
**Figure 2.9 WebRTC video chat**

## 2.10 Website Development like Bark.com

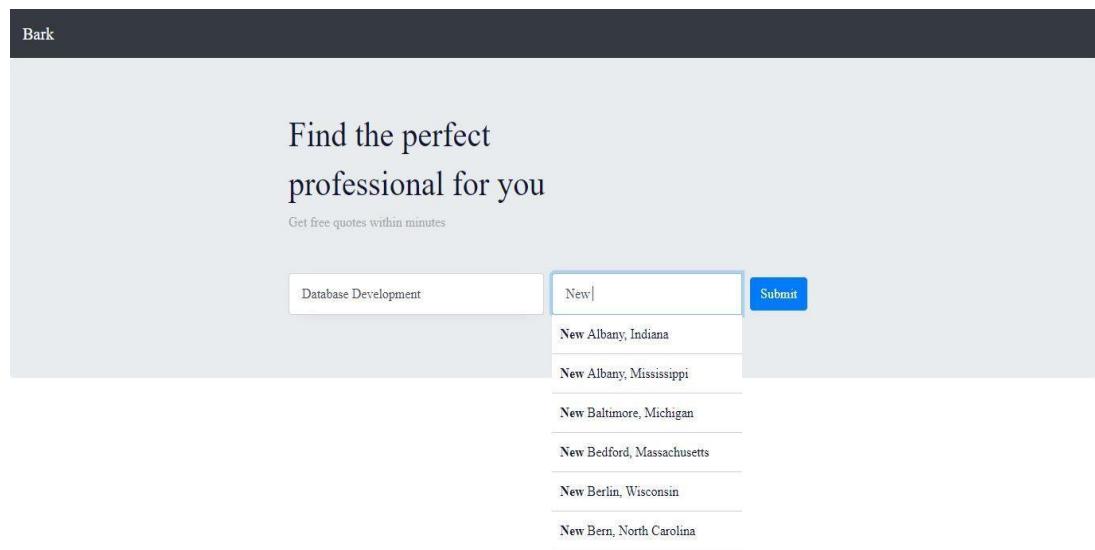
Bark.com instantly finds the best local service professionals for you - fast & free. Save time & money and get your job done fast. So we were supposed in this training activity to develop the frontend UI template and make the core functionality works in backend by using the API for location and services with respective frequent questions, also to store the user response in the Database MYSQL using the API. Furthermore, the website is still under development phase.

### 2.10.1 UI Template Frontend Development of the website

The UI was designed using the HTML, CSS and, Bootstrap and JavaScript and screenshots of the template as shown below

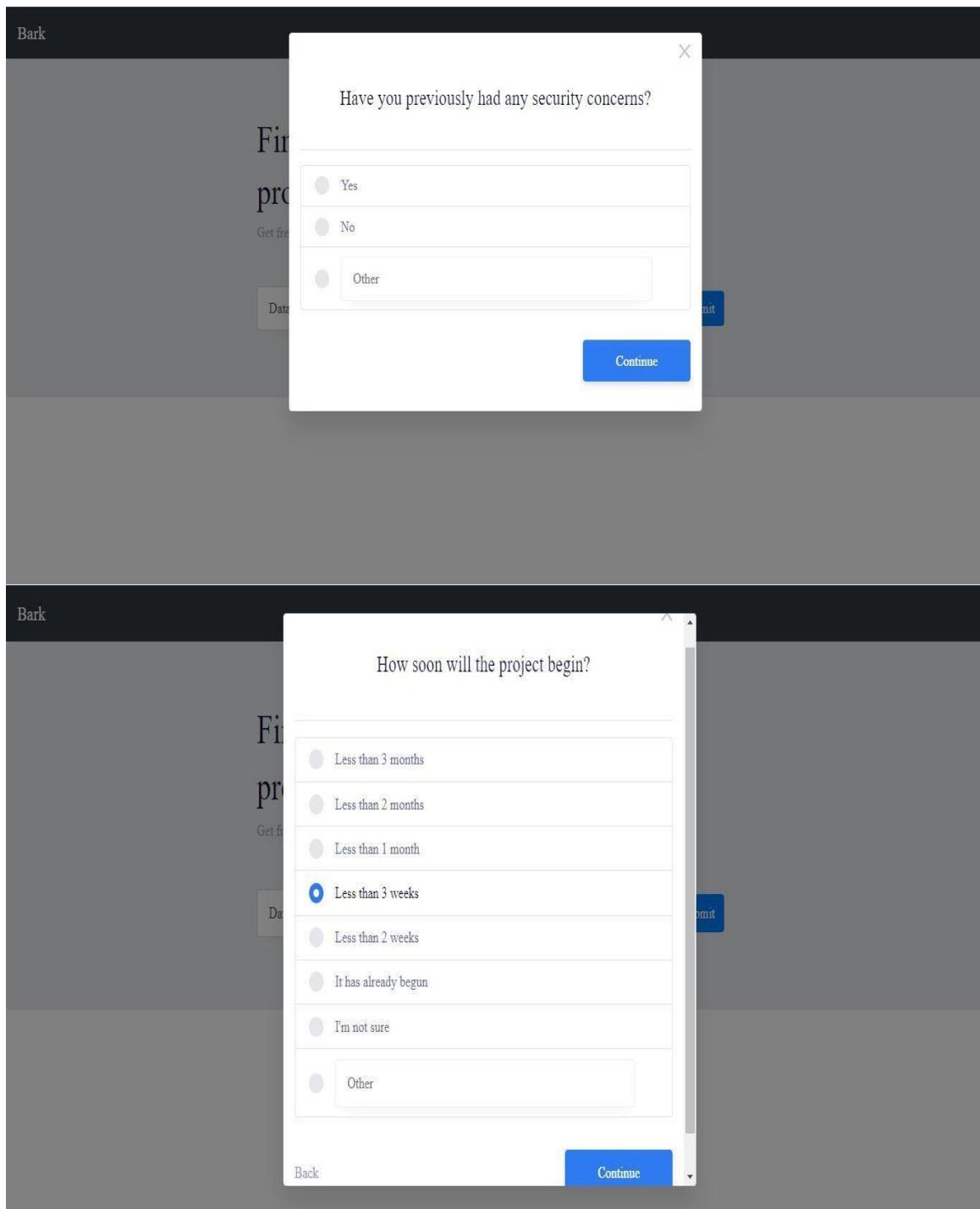


The screenshot shows the Bark.com homepage. At the top, there's a dark header bar with the word "Bark". Below it, the main content area has a light gray background. In the center, the text "Find the perfect professional for you" is displayed above a subtext "Get free quotes within minutes". Below this, there are two input fields: one for "Service" containing "Data" and another for "Location" containing "New". To the right of these fields is a blue "Submit" button. Below the input fields, there are three horizontal lines of service categories: "Database Security", "Database Development", and "Data Recovery".

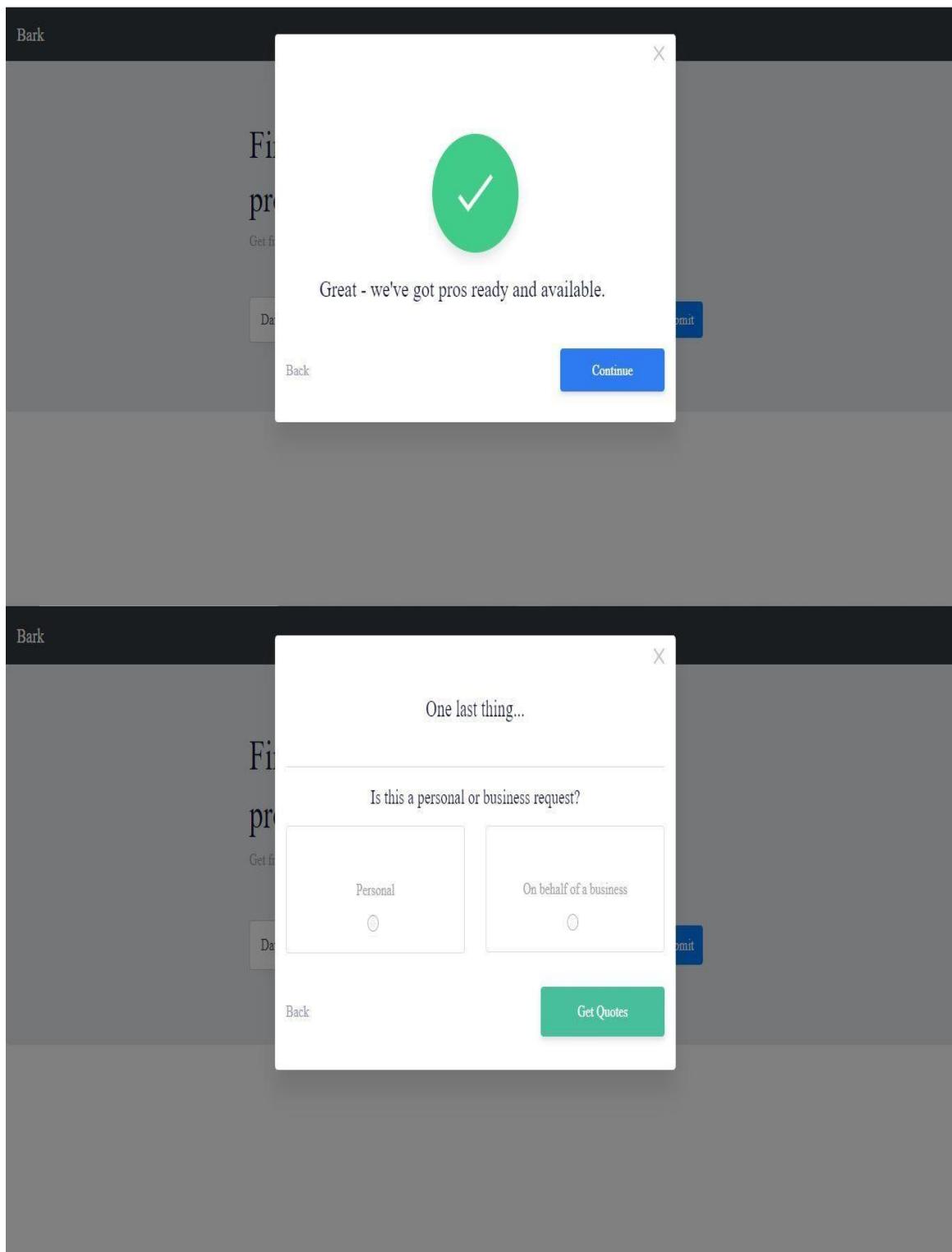
**Figure 2.10.1 (a) Homepage Display the Services**


This screenshot shows the same Bark.com homepage as the previous one, but with a different state of the "Location" input field. The input field now contains "New" and has a dropdown menu open, listing several location suggestions: "New Albany, Indiana", "New Albany, Mississippi", "New Baltimore, Michigan", "New Bedford, Massachusetts", "New Berlin, Wisconsin", and "New Bern, North Carolina". The "Submit" button is visible to the right of the input field.

**Figure 2.10.1 (b) Homepage Display the Locations**



**Figure 2.10.1 (c) Frequent Questions with respective Services**

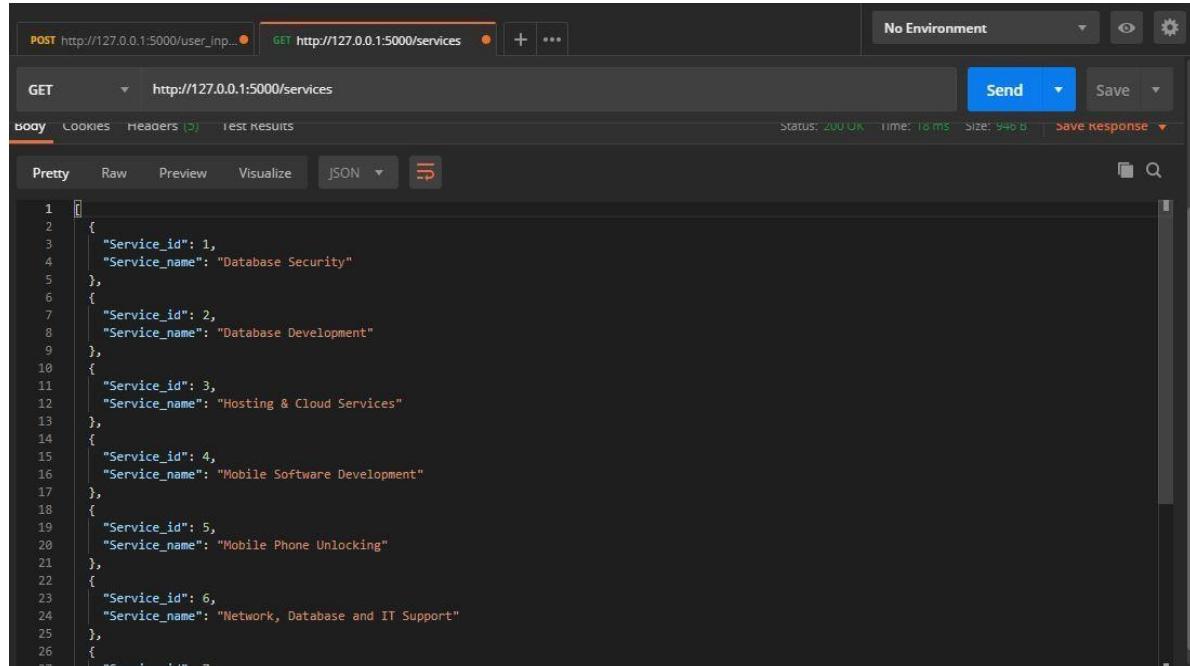


**Figure 2.10.1 (d) Final Response message and Request Question to User**

### 2.10.2 Backend Development with MYSQL Database and API

We have developed the API for core functionality working such as for the location and the various services as well as to store the user input to MYSQL Database.

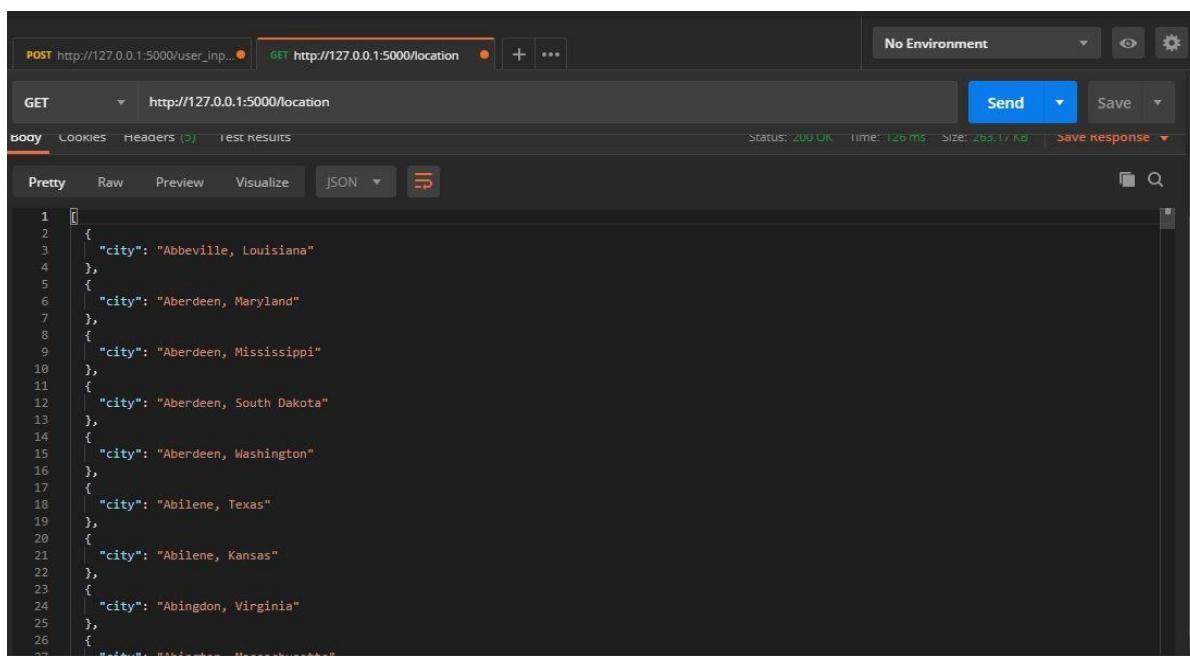
API developed for the location and various services loads JSON data when it receives API call request as a result response all the service and location are loaded. Also API for user input stores the final data in MYSQL Database.



```

POST http://127.0.0.1:5000/user_inp... | GET http://127.0.0.1:5000/services | + ...
No Environment | Send | Save | ...
GET http://127.0.0.1:5000/services | Status: 200 OK | Time: 18 ms | Size: 948 B | Save Response ...
Body Cookies Headers (2) Test Results | Pretty Raw Preview Visualize JSON ...
Pretty Raw Preview Visualize JSON ...
1 [
2   {
3     "Service_id": 1,
4     "Service_name": "Database Security"
5   },
6   {
7     "Service_id": 2,
8     "Service_name": "Database Development"
9   },
10  {
11    "Service_id": 3,
12    "Service_name": "Hosting & Cloud Services"
13  },
14  {
15    "Service_id": 4,
16    "Service_name": "Mobile Software Development"
17  },
18  {
19    "Service_id": 5,
20    "Service_name": "Mobile Phone Unlocking"
21  },
22  {
23    "Service_id": 6,
24    "Service_name": "Network, Database and IT Support"
25  },
26  {
27    "Service_id": 7,
28    "Service_name": "System Security"
29  }
30 ]
  
```

Figure 2.10.2 (a) API for various Services



```

POST http://127.0.0.1:5000/user_inp... | GET http://127.0.0.1:5000/location | + ...
No Environment | Send | Save | ...
GET http://127.0.0.1:5000/location | Status: 200 OK | Time: 126 ms | Size: 165.17 KB | Save Response ...
Body Cookies Headers (2) Test Results | Pretty Raw Preview Visualize JSON ...
Pretty Raw Preview Visualize JSON ...
1 [
2   {
3     "city": "Abbeville, Louisiana"
4   },
5   {
6     "city": "Aberdeen, Maryland"
7   },
8   {
9     "city": "Aberdeen, Mississippi"
10  },
11  {
12    "city": "Aberdeen, South Dakota"
13  },
14  {
15    "city": "Aberdeen, Washington"
16  },
17  {
18    "city": "Abilene, Texas"
19  },
20  {
21    "city": "Abilene, Kansas"
22  },
23  {
24    "city": "Abingdon, Virginia"
25  },
26  {
27    "city": "Adrian, Michigan"
28  }
29 ]
  
```

Figure 2.10.2 (b) API for Locations

The screenshot shows the Postman application interface. At the top, it displays a POST request to "http://127.0.0.1:5000/user\_input". The status bar indicates "No Environment". Below the request, the "Body" tab is selected, showing a JSON payload with 14 lines of code. The response tab shows a successful "201 CREATED" status with a response body identical to the request. The bottom right corner shows the "Save Response" button.

```

1+ {
2  "Q1": "yes",
3  "Q2": "bye",
4  "Q3": "less-than-3-months",
5  "Q4": "up-to-6-months",
6  "Q5": "personal-requirement",
7  "Q6": "Anuj",
8  "Service_name": "Database Security",
9  "email_id": "adev23452@gmail.com",
10 "id": 1,
11 "name": "Anuj Dev",
12 "phone_no": "9",
13 "location": "california1"
14 }

```

Figure 2.10.2 (c) API for storing the User Response to MYSQL Database

The screenshot shows the MySQL Workbench interface with a query editor containing the SQL command "select \* from userinfo.user\_table". Below the editor is a results grid titled "Result Grid" showing data from the user\_table. The columns are service\_name, location, Q1, Q2, Q3, Q4, Q5, Q6, and email\_id. The data includes various entries such as "Database Security" at "Alamo Heights, Texas" and "Hosting & Cloud Services" at "Aberdeen, Maryland". The results grid has a toolbar with icons for filtering, exporting, and wrapping cell content. On the right side, there are tabs for "Result Grid", "Form Editor", and "Field Types".

service_name	location	Q1	Q2	Q3	Q4	Q5	Q6	email_id
Database Security	Alamo Heights, Texas	yes	no	Less than 1 month	Up to 1 year	Medium	msg	johngreen@gmail.com
Hosting & Cloud Services	Aberdeen, Maryland	None	None	None	None	None	None	None
Hosting & Cloud Services	Aberdeen, Maryland	no	yes	Less than 2 months	Over 1 year	Small	message	marrygreen@gmail.com
Database Security	california	yes	bye	less-than-3-months	up-to-6-months	personal-requirement	Anuj	adev23452@gmail.com
Database Security	Caldwell, New Jersey	no	yes	less-than-3-weeks	up-to-1-month	personal-requirement	a	a
Database Security	california1	yes	bye	less-than-3-months	up-to-3-months	personal-requirement	Anuj	adev23452@gmail.com
Database Security	Babylon, New York	yes	no	less-than-2-months	up-to-3-months	personal-requirement	ds	cc
Database Security	Cabot, Arkansas	no	no	less-than-3-weeks	up-to-3-months	personal-requirement	wdsadsdsada	harshi@gmail.com
Database Security	california1	yes	bye	less-than-3-months	up-to-6-months	personal-requirement	Anuj	adev23452@gmail.com

Figure 2.10.2 (d) MYSQL Database stores User Response

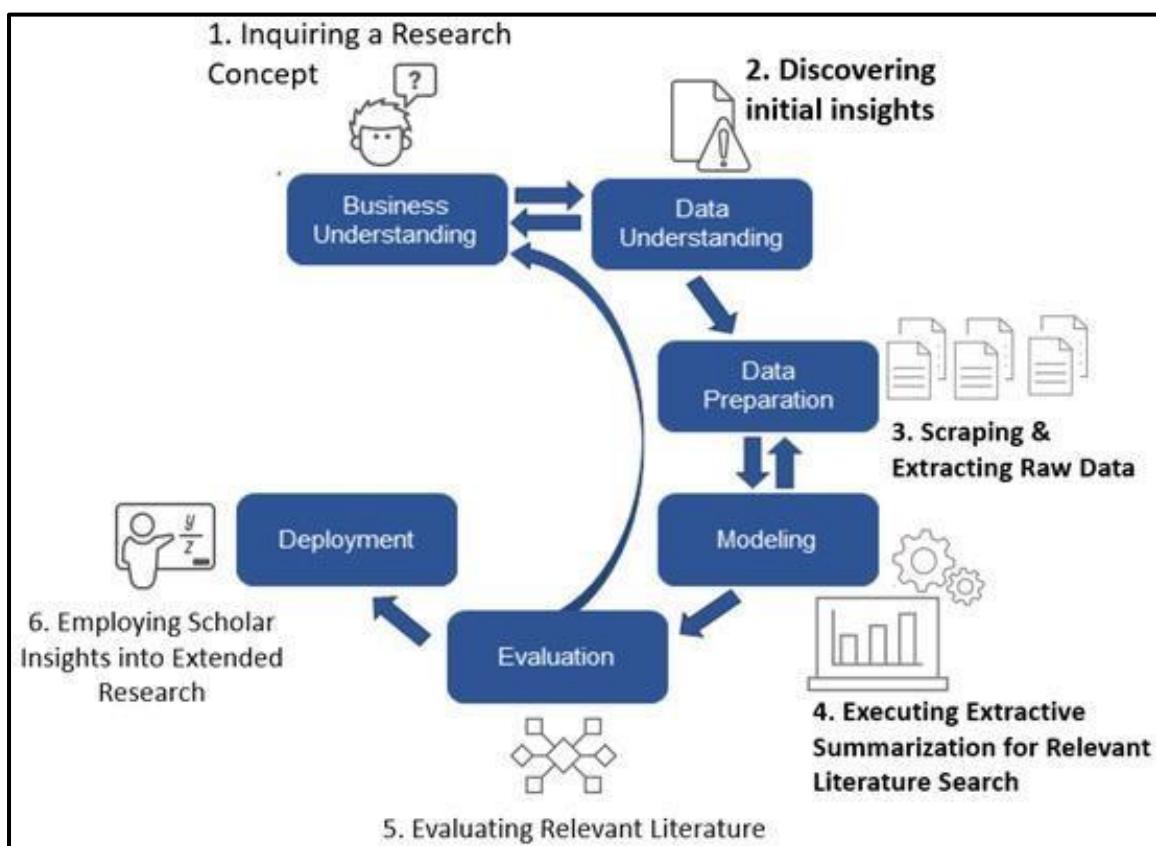
## CHAPTER 3 SYSTEM PLANNING

### 3.1 Web Scraping with Python of 4200+ car dealers

#### 3.1.1 System planning and software model for web scraping

CRISP-DM (cross-industry standard process for data mining) is a robust and well-proven methodology that provides a structured approach to solve virtually any analytics problem in any industry. It provides anyone from novices to data mining experts with a complete blueprint for conducting a data mining project.

Process of CRISP-DM framework:



**Figure 3.1.1 CRISP- DM model lifecycle**

Advantages of the model:

- CRISP-DM framework provides a uniform framework for Guidelines and experience documentation
- CRISP-DM is flexible to account for different business/agency problems and different data.

### 3.1.2 Functional Requirements (Software Requirement Specification)

**Table 3.1.2 Functional Requirements**

Sr.no	Name of Functional Requirements	Description
FR#0	Jupyter Notebook Spyder	It should be an integral part of any Python data scientist's toolbox. It's great for prototyping and sharing notebooks with visualizations
FR#1	Python language	Python is an interpreted, highlevel, general-purpose language
FR#2	Python language packages and libraries	<b>BeautifulSoup4</b> for handling all of the HTML processing and <b>requests</b> for performing your HTTP requests

### 3.1.3 Modules of the system

We were given the client project initially to scrap the address and contact details of the sample data containing the 80 car dealers and after that the main project of scraping the reviews and ratings of 4200+ car dealers from different websites and both of the modules as below

#### 3.1.3.1 Module-1:-Web Scraping with Python Beautiful Soap

In the sample data CSV file where it contains the 80+ car-dealer names and we have done the Address and Contact Scraping and then we were provided with the actual data which contains the 4200+ car dealers details and we need to do the ratings and review scraping from different sites such as Google, Facebook, BBB.org, Yelp, Cars.com and dealer rater where we used the Beautiful Soap and requests library in Python and HTML parsers.

#### 3.1.3.2 Module-2:-Data Selection

In both the above-mentioned projects after scraping the required data and storing them at appropriate column values in the CSV file the next step includes the selection of the data which are useful in data mining and analysis purpose.

### **3.1.3.3 Module-3:-Data Pre-processing and Tidying**

In this module of the project after the data selection, we found that many of the data are missing, having null values, some includes the inconsistent data which points to the noisy data or incorrect values than such values must be removed is carried in this process which makes the data more accurate.

### **3.1.3.4 Module-4:-Data Transformation**

In this module the processing of data refers to its transformation where we have removed all the unwanted data and further transforming data to make it precisely work with the mining process and the analysis purpose for future use, We have removed here the unwanted and missing values found in google and yelp and repeated values are also replaced with the correct values on the same.

### **3.1.3.5 Module-5:-Data Mining**

In this module, all the data in CSV files are accurate and without any errors on that where we have applied the data mining process by recognizing various patterns in reviews and ratings and address and contact details.

### **3.1.3.6 Module-6:-Interpretation**

The final step after all the processing includes the interpretation of data in an understandable format to all the viewers including the technical and non-technical where we formed the visualization diagrams and the format patterns.

## **3.2 COVID-19 Patient Predictor System with Machine Learning**

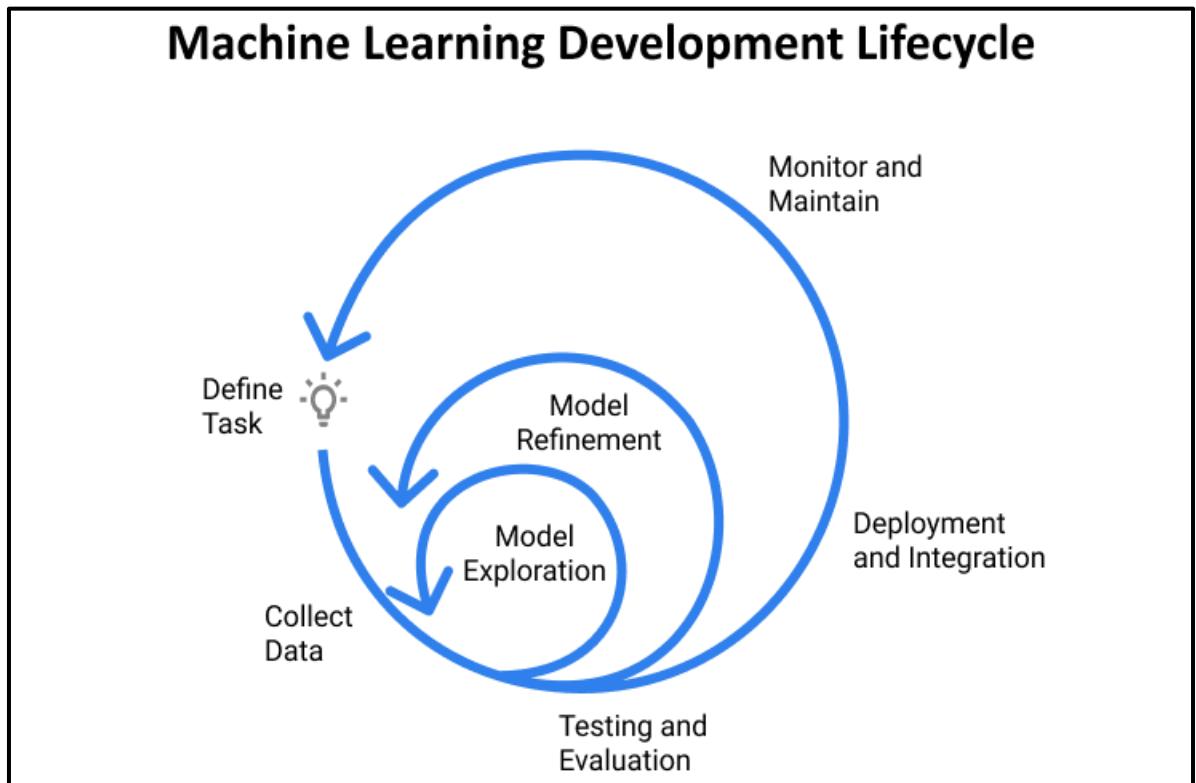
### **3.2.1 System planning and model for the machine learning**

Machine learning has given the computer systems the ability to automatically learn without being explicitly programmed. But how does a machine learning system work? So, it can be

described using the life cycle of machine learning. The machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project.

Machine learning life cycle involves seven major steps:

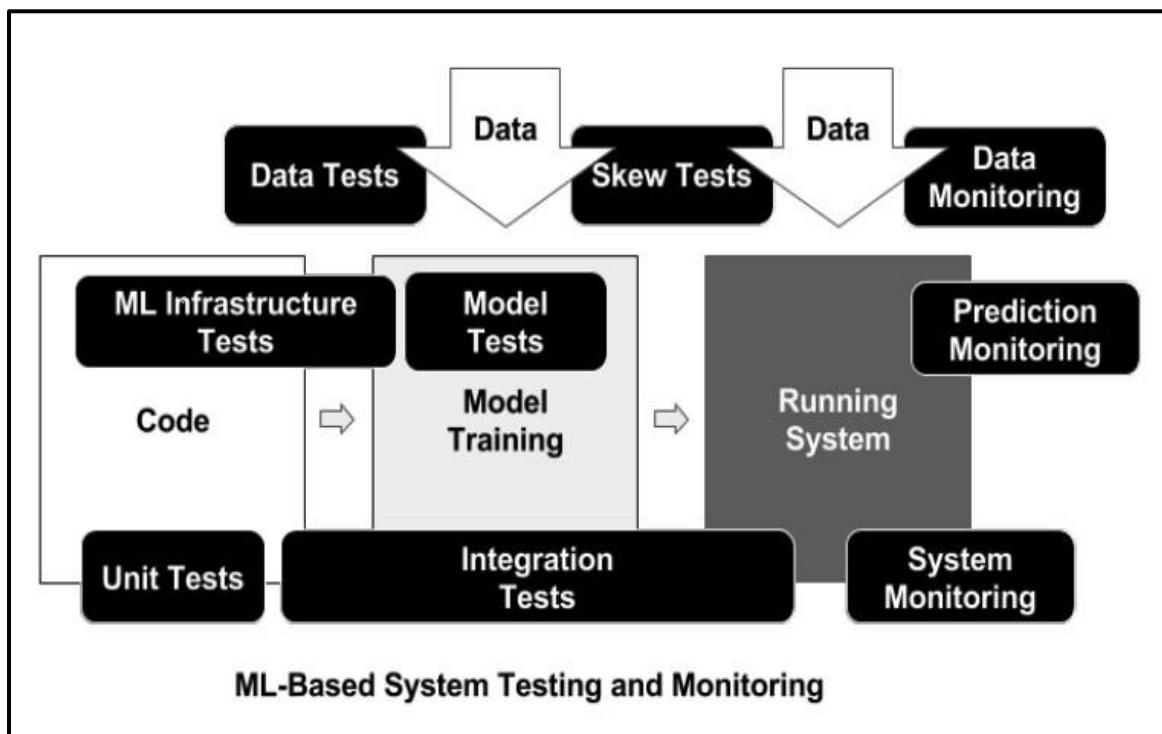
- **Gathering Data, Data preparation, Data Wrangling, Analyse Data, Train the model, Test the model, Deployment**



**Figure 3.2.1 (a) Machine Learning Deployment Cycle**

The figure depicts tests implemented throughout the development of the model such as data quality tests, model performance tests, and machine learning infrastructure tests. None of these tests, however, will be the focus of this blog post, though they are very important as well. The testing and monitoring processes need to continue running even after full deployment to ensure that the model continues providing value even after some time has

passed. Compared to software testing, testing machine learning models in deployment is an underdeveloped area of exploration. Most organizations don't understand what to test exactly to truly validate a model



**Figure 3.2.1 (b) Testing and Monitoring Cycle**

Advantages of Machine Learning model:-

The machine learning life cycle is important because it delineates the role of every person in a company in data science initiatives, ranging from business to engineering personnel. It takes every project from inception to completion and gives a high-level perspective of how an entire data science project should be structured in order to result in real, practical business value. Failing to accurately execute on any one of these steps will result in misleading insights or models with no practical value.

### 3.2.2 Functional Requirements (Software Requirement Specification)

**Table 3.2.2 Functional Requirements**

Sr.no	Name of Functional Requirements	Description

FR#0	Jupyter Notebook  Spyder	It should be an integral part of any Python data scientist's toolbox. It's great for prototyping and sharing notebooks with visualizations
FR#1	Python language	Python is an interpreted, highlevel, general-purpose language
FR#2	Python language packages and libraries	Various machine learning python library such as numpy, pandas, scipy, Keras, etc
FR#3	Python Flask	Flask is a microframework in python for designing a web app.

### 3.2.3 Modules of the COVID-19 Patient Predictor System

Machine learning model which predicts if a person can have a probability or not of having COVID-19. With this, the person can self-assess and don't panic and be at home. It can also help prioritize the patients who really have Coronavirus and help the health care workers who really have it and save kits for the needed.

#### 3.2.3.1 Module-1:- Gathering of COVID-19 Symptoms

After referring various sites and tutorials we gathered the symptoms for the project as in machine learning the preparation of data is the first and most important step and we listed the following:- Age, Fever, Fatigue, Runny Nose, Difficulty in breathing and Infection probability act as our project parameters and created the CSV file format for the data.

#### 3.2.3.2 Module-2: - Model Training and Requirement

The process of training an ML model involves providing an ML algorithm (that is, the learning algorithm) with training data to learn from hence in this module we started to write our algorithm with the python based on above-selected parameters in python. The training data must contain the correct answer, which is known as a target or target attribute and our targeted attribute is Infection probability where 1 is for yes and 0 is no. Our learning algorithm finds patterns in the training data that map the input data attributes to the target

and it outputs an ML model that captures these patterns which help in the prediction of COVID-19 symptoms in the patient.

### **3.2.3.3 Module-3:- Webapp with Flask**

Flask is a Python-based microframework used for developing small scale websites. Flask is very easy to make Restful API's using python. It includes various HTML and CSS files for designing the web app

In this module, we have designed the web app with a microframework of Python Flask. This web app asks for the input to the user and finally displays the probability of a patient being affected by the patient due to coronavirus.

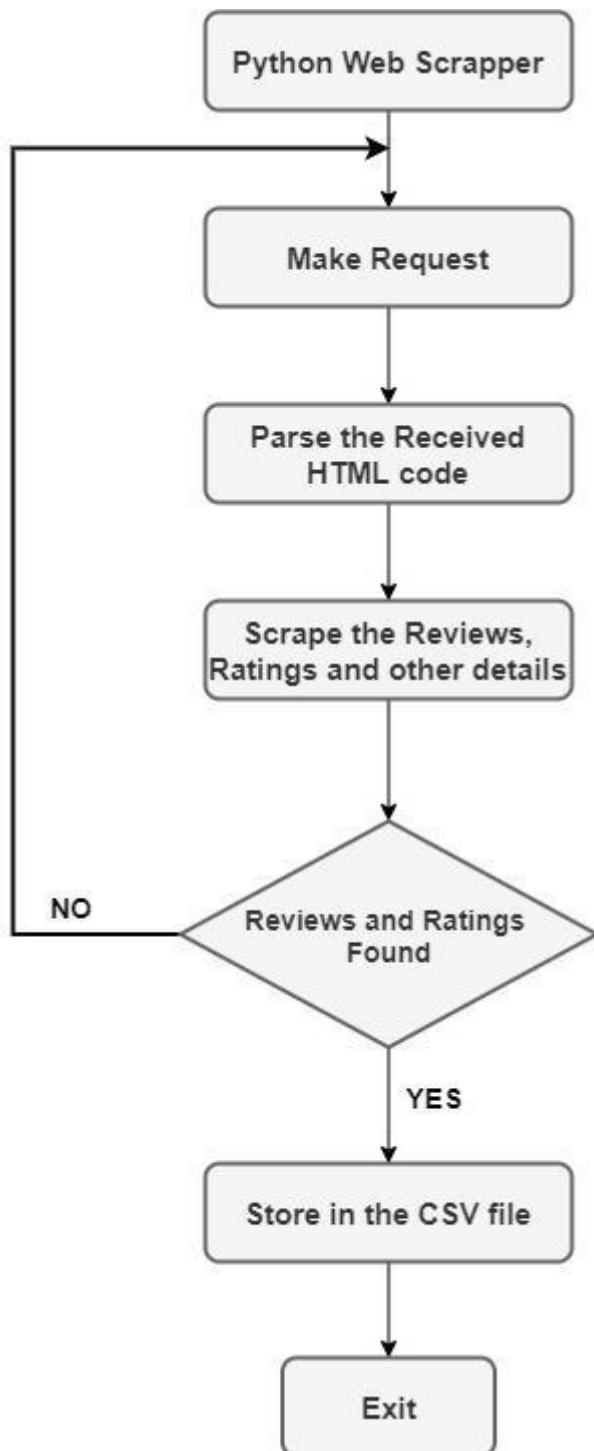
### **3.2.3.4 Module-4:- Deploying the machine learning model to the flask**

In this module, we have integrated the Flask app with our machine learning model and finally implemented the system with good accuracy algorithm. Also, we have hosted our web app on AWS ec2 instance.

## CHAPTER 4 SYSTEM DESIGN

### 4.1 Web Scraping with Python of 4200+ car dealers

#### 4.1.1 System Process Flow Diagram

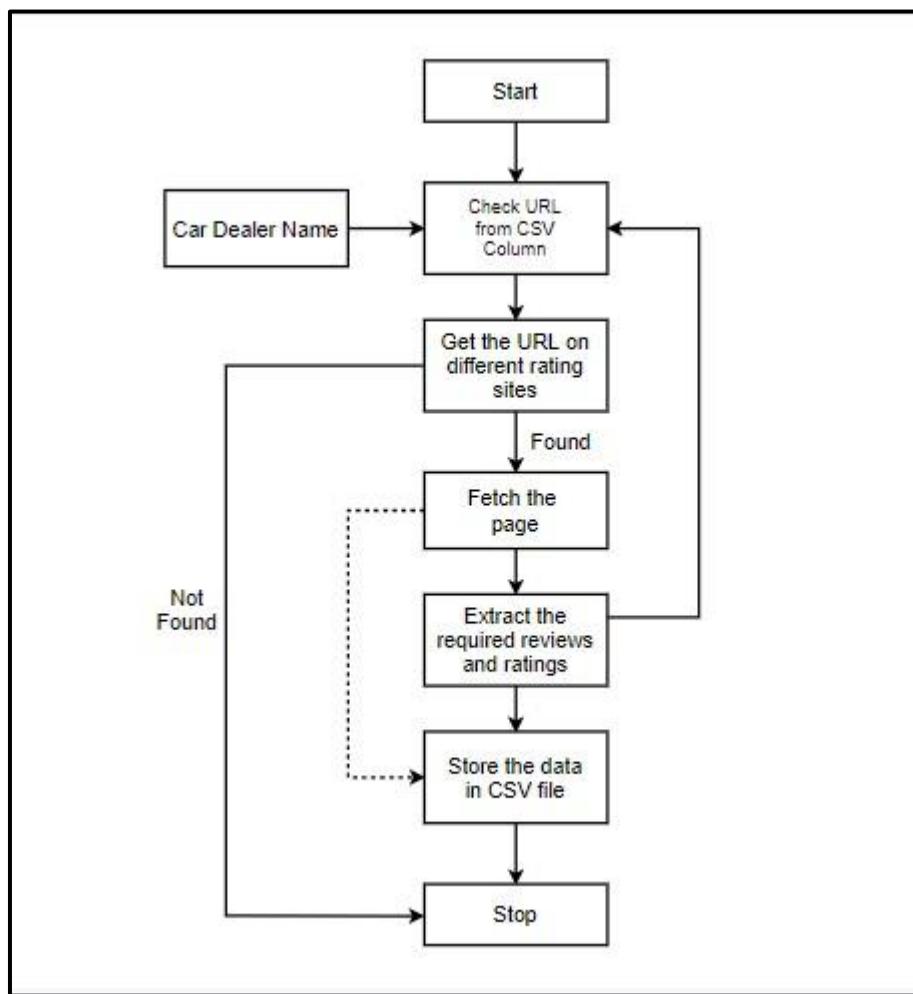


**Figure 4.1.1 System Process Flow Diagram**

#### 4.1.2 System Workflow Diagram

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The below flow diagram represents the working of the web scrapper of python that checks URL using the car dealer name and then gets the required details of review and rating and stores it in the CSV format.

**Figure 4.1.2 System Workflow Diagram**

#### 4.1.3 Activity Diagram

An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system. The below activity diagram portrays the control flow from a start point to a finish point showing the various

decision paths that exist while the activity is being executed for our web scraping system with python.

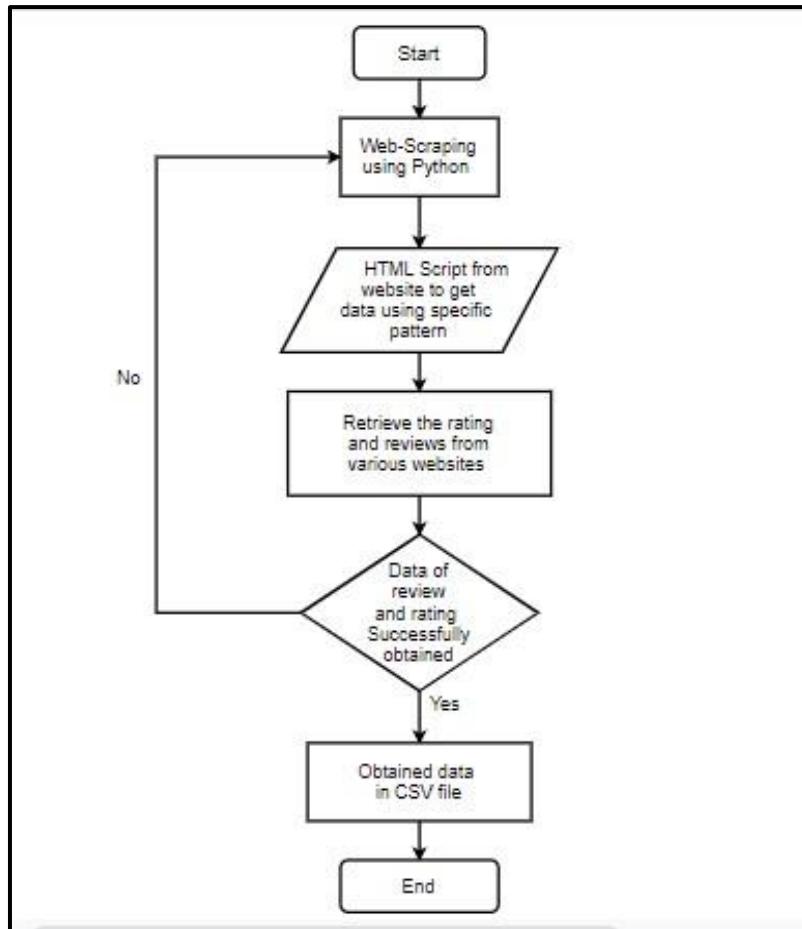
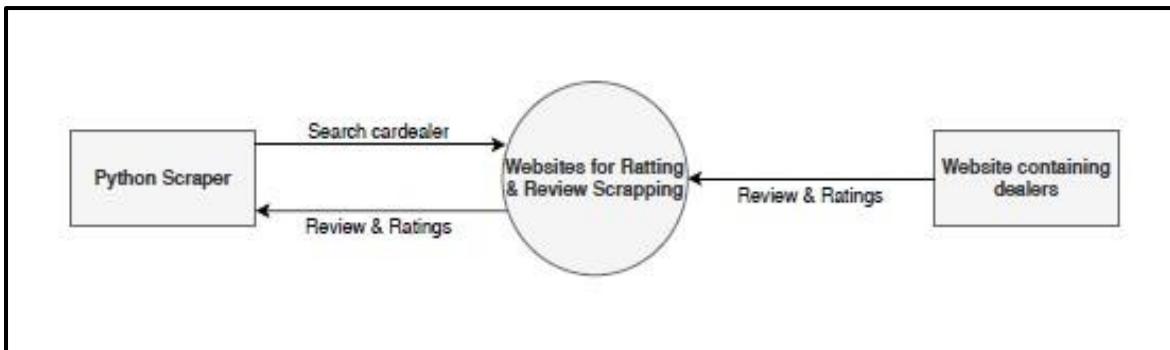


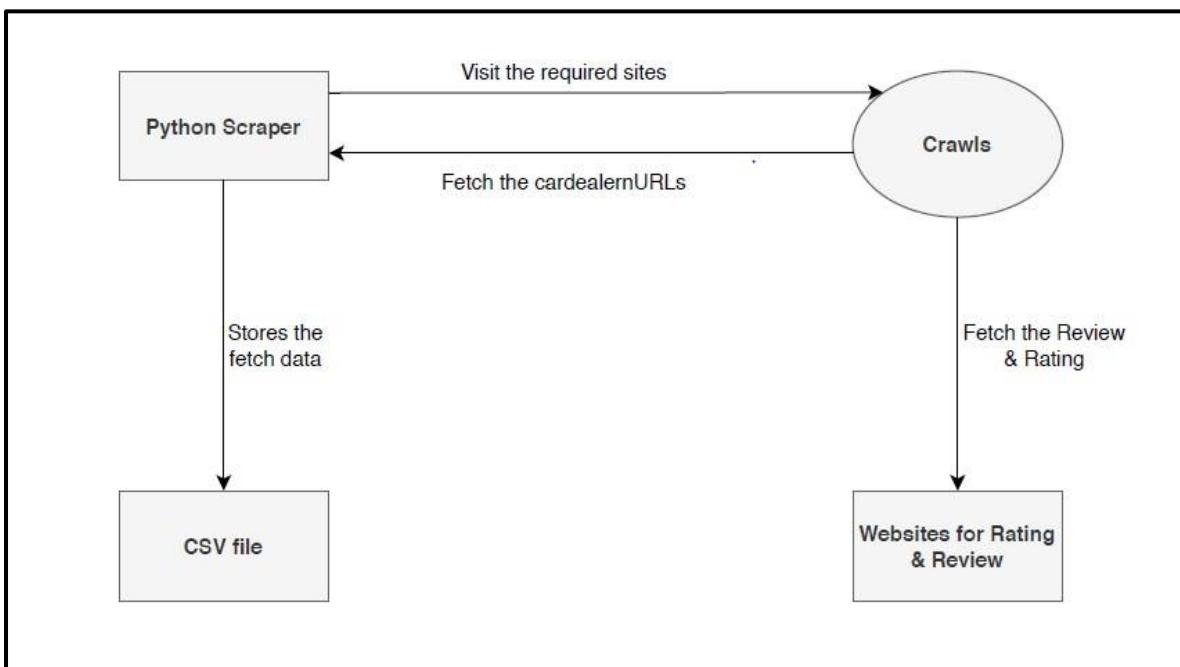
Figure 4.1.3 Activity Diagram

#### 4.1.4 Data Flow Diagram (DFD)

A data-flow diagram is a way of representing a flow of data of a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops.

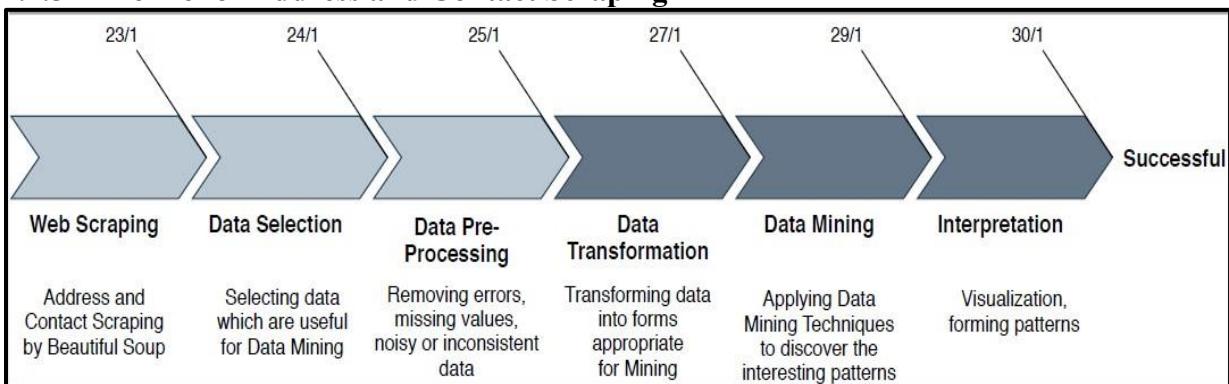


**Figure 4.1.4 (a) DFD LEVEL-0**



**Figure 4.1.4 (b) DFD LEVEL-1**

#### 4.1.5 Timeline for Address and Contact Scraping



**Figure 4.1.5 Timeline chart**

#### 4.1.6 Timeline for Reviews and Rating Scarping

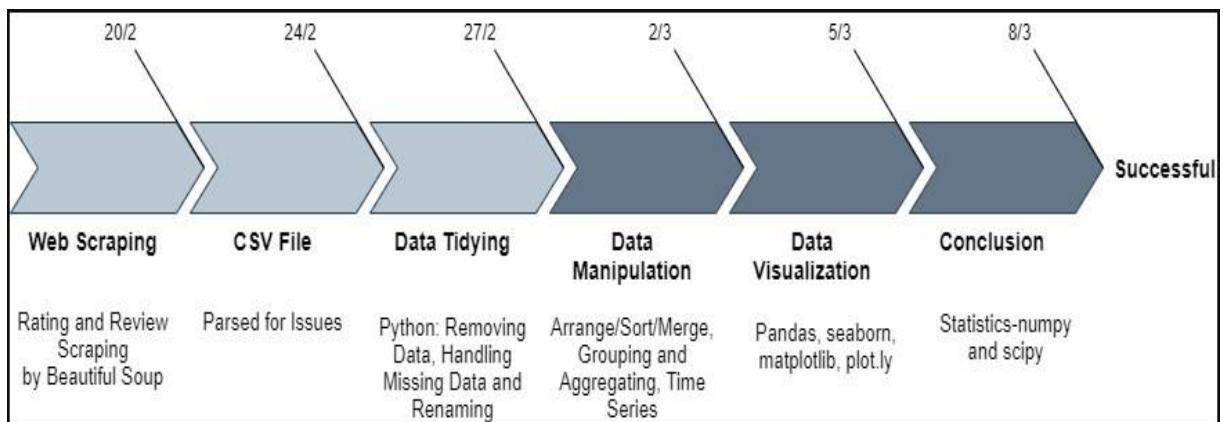


Figure 4.1.6 Timeline chart

#### 4.2 COVID-19 Patient Predictor System with Machine Learning

##### 4.2.1 Timeline Chart

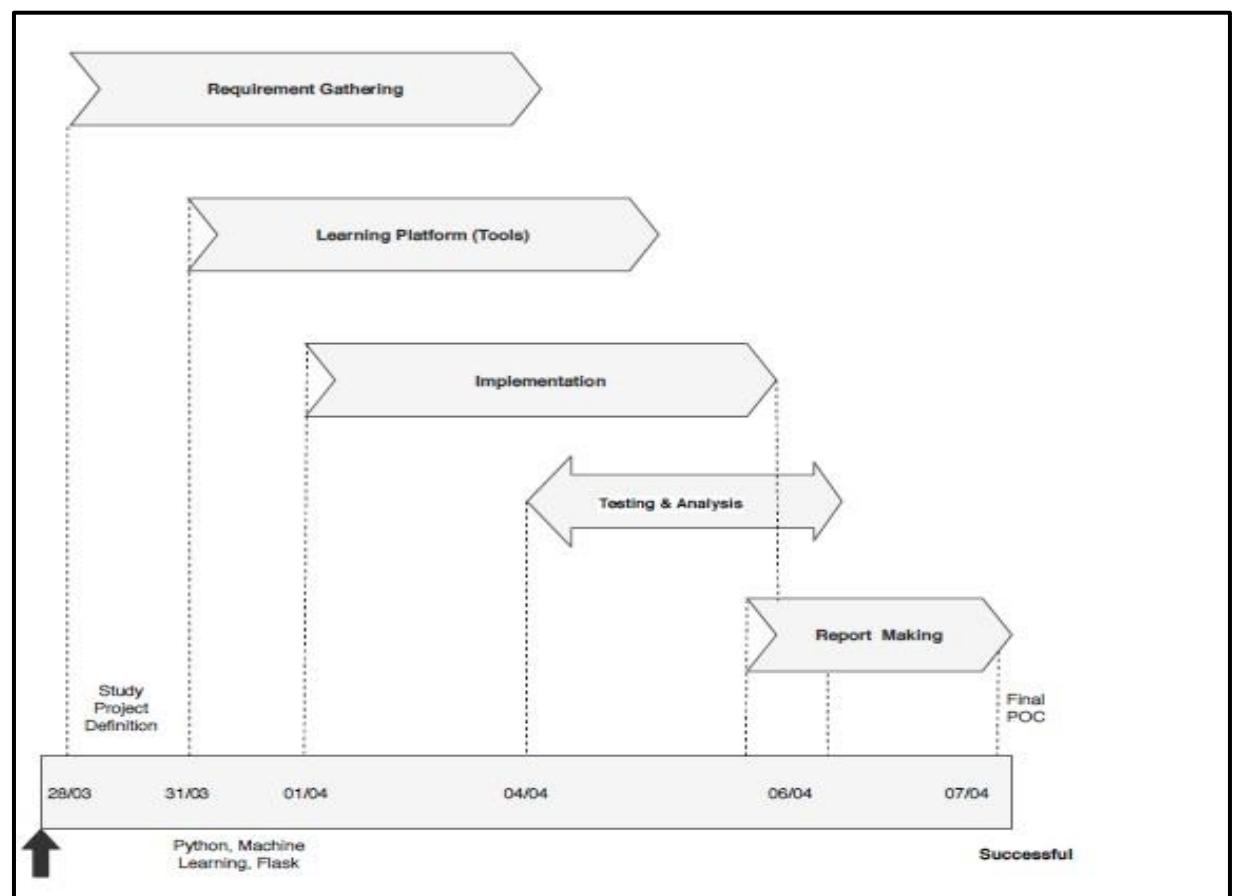


Figure 4.2.1 Timeline chart

#### 4.2.2 Sequential Workflow Diagram

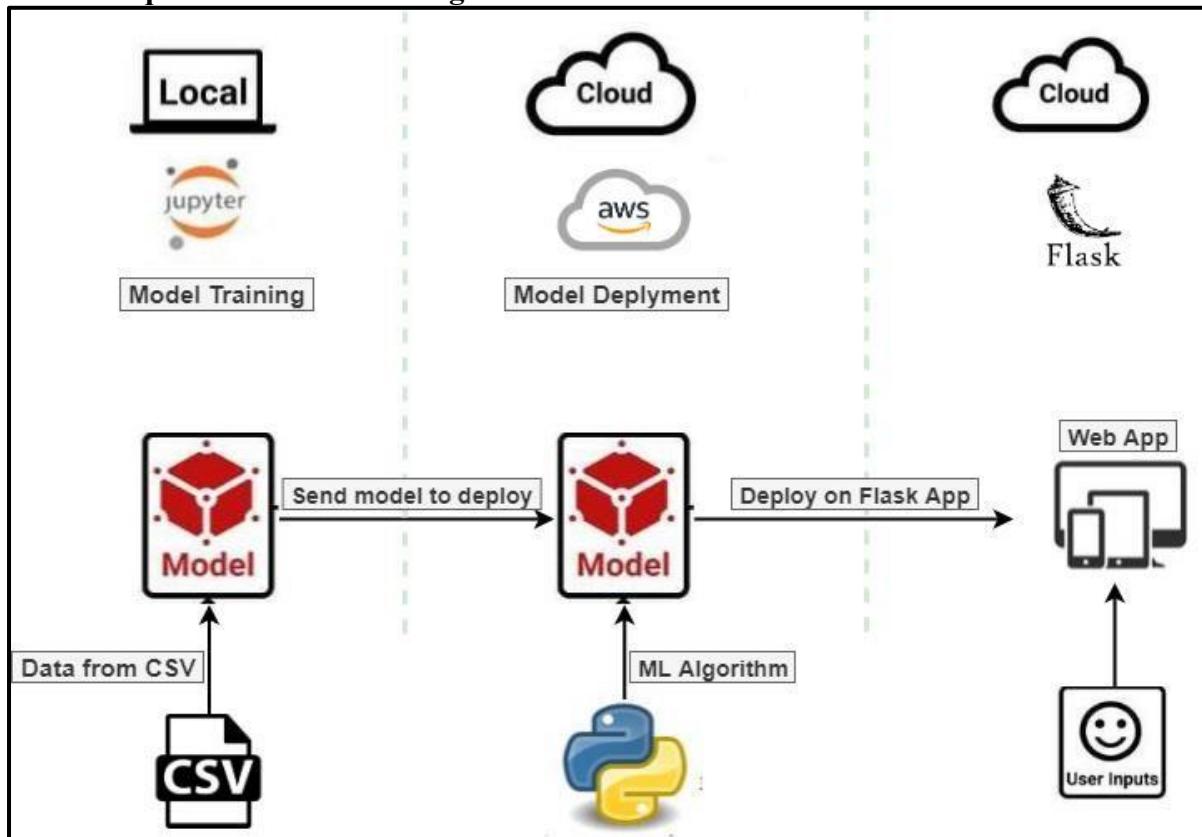


Figure 4.2.2 Sequential Workflow Diagram

#### 4.2.3 System Architecture Diagram

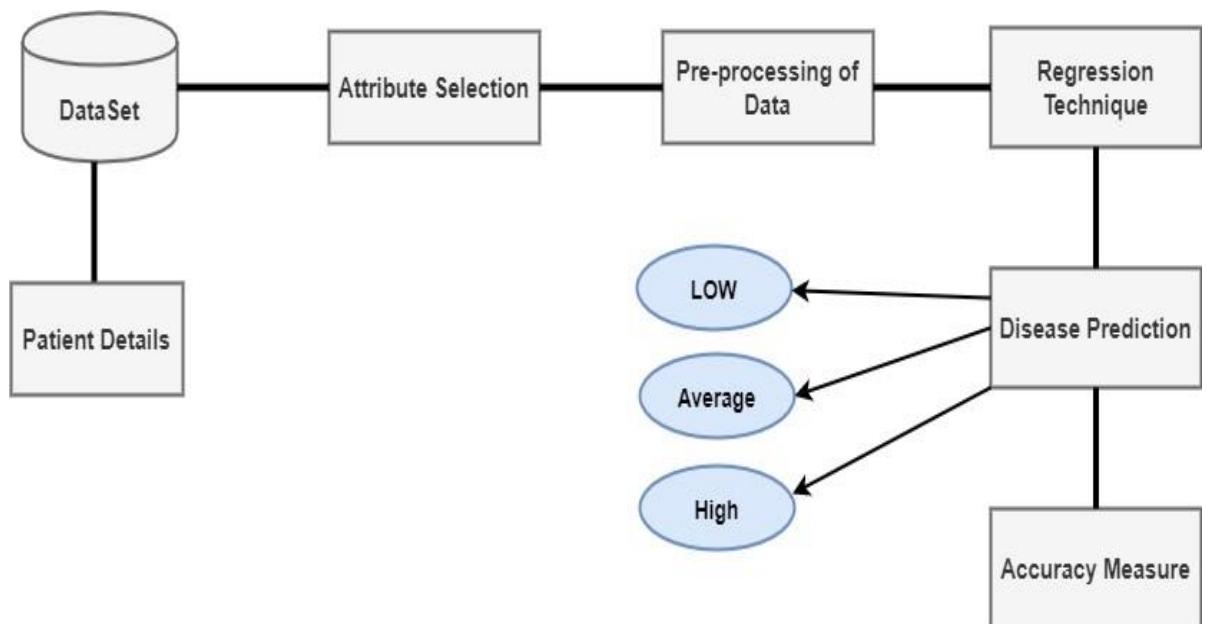


Figure 4.2.3 System Architecture

#### 4.2.4 System Processing Workflow Diagram

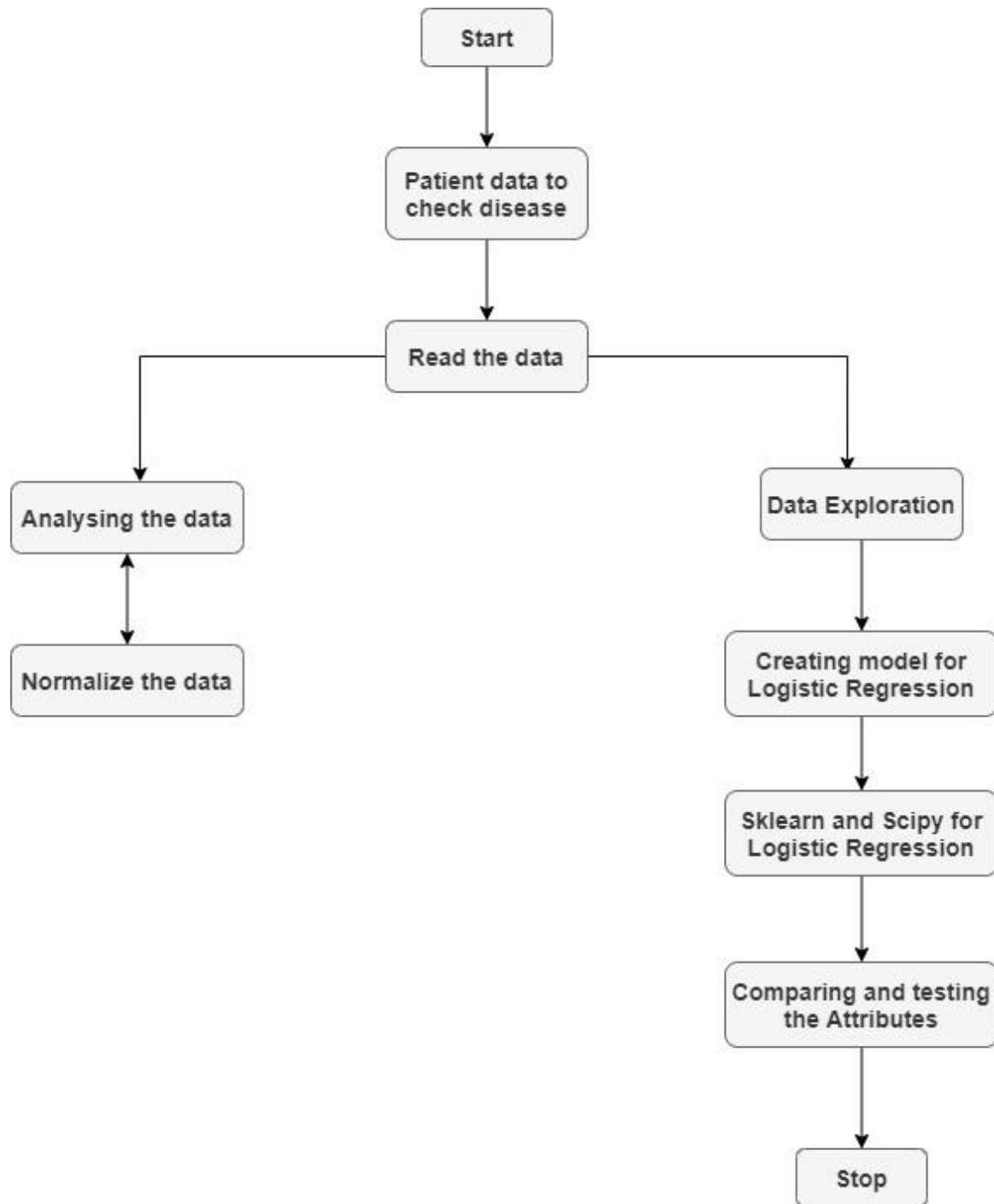


Figure 4.2.4 Processing Workflow Diagram

## 4.3 AWS (Amazon Web Services)

### 4.3.1 Timeline Chart

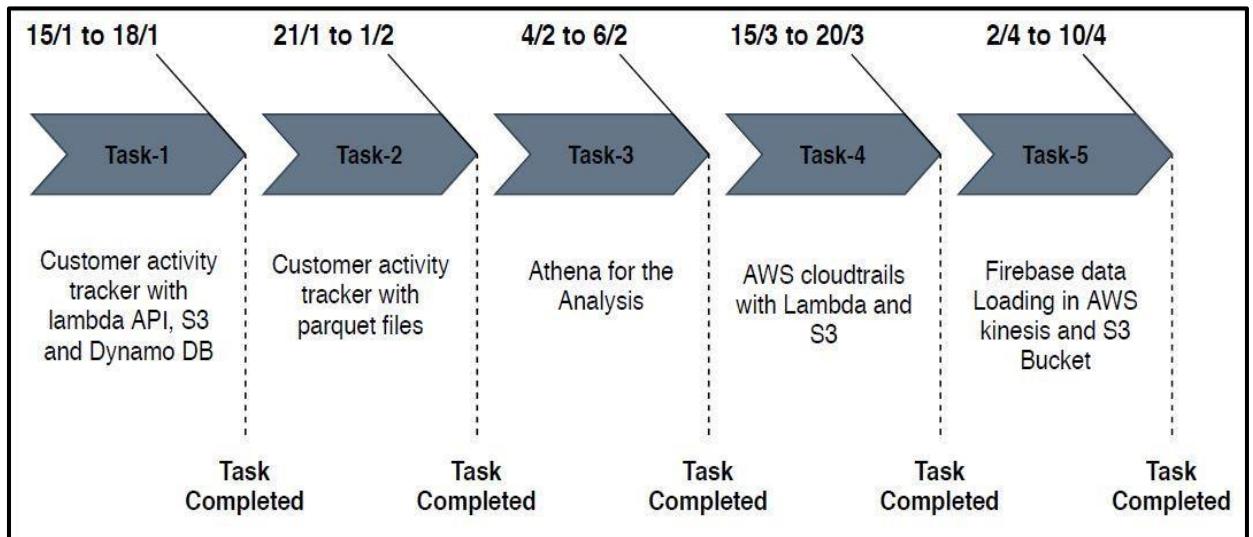


Figure 4.3.1 Timeline chart

## CHAPTER 5 IMPLEMENTATION AND TESTING

### 5.1 Web Scraping with Python of 4200+ car dealers

#### 5.1.1 Software Requirements

- Python language and its libraries such as beautiful soap for HTML parser to scrape data from websites and requests module for HTTP request and panda's library to deal with CSV as a data frame.
- Jupyter notebook for testing and running the scrapper

## 5.1.2 Snapshots

### ○ CSV data file from Client

The screenshot shows an Excel spreadsheet with two main tables. The top table has columns for City, Count, City, State, Brand, Dealerships Name, Address, City, State, Postcode, Phone Number, and Website. The bottom table has columns for Google Review Score, Number of Google Reviews, YELP Review Scores, Number of Yelp Reviews, Cars.com Review Scores, Number of Cars.com Reviews, DealerRater Review Scores, Number of DealerRater Reviews, FaceBook Review Scores, Number of Facebook Reviews, and BBB.org Number of Complaints (3 Year).

	B	C	D	E	F	G	H	I	J	K	L	
	CITY	COUNT	CITY	STATE	BRAND	Dealerships Name	Address	CITY	State	Postcode	Phone Number	Website
BUICK GMC	5	VA BEACH	VA	ACURA	Hall Acura Newport News	12501 Jefferson Ave	Newport News	VA	23602	757-886-7060	<a href="https://www.hallacura.com">https://www.hallacura.com</a>	
CHEVROLET	10	VA BEACH	VA	ACURA	Hall Acura Virginia Beach	3200 Virginia Beach Blvd	Virginia Beach	VA	23452	757-631-3060	<a href="https://www.hallacura.com">https://www.hallacura.com</a>	
CDR JEEP	8	VA BEACH	VA	ACURA	Priority Acura	1701 S Military Hwy	Chesapeake	VA	23320	757-420-5114	<a href="https://www.priorityacura.com">https://www.priorityacura.com</a>	
FORD	10	VA BEACH	VA	AUDI	Audi Hampton	2712 Magruder Blvd,	Hampton	VA	23666	(888) 860-5597	<a href="https://www.audihampton.com">https://www.audihampton.com</a>	
HONDA	6	VA BEACH	VA	AUDI	Audi Virginia Beach	2865 Virginia Beach Blvd,	Virginia Beach	VA	23452	(888) 883-5329	<a href="https://www.audivirginiatech.com">https://www.audivirginiatech.com</a>	
HYUNDAI	6	VA BEACH	VA	BMW	Checkered Flag BMW	5225 Virginia Beach Blvd	Virginia Beach	VA	23462	(757) 490-1111	<a href="https://www.checkeredflagbmw.com">https://www.checkeredflagbmw.com</a>	
MAZDA	3	VA BEACH	VA	BMW	Casey BMW	12861 Jefferson Ave	Newport News	VA	23608	(757) 989-1300	<a href="https://www.caseybmw.com">https://www.caseybmw.com</a>	
TOYOTA	7	VA BEACH	VA	BUICK GMC	DUKE AUTOMOTIVE CORP.	2016 N MAIN ST	SUFFOLK	VA	23434	(757) 539-8777	<a href="https://www.dukeautomotive.com">https://www.dukeautomotive.com</a>	
NISSAN	7	VA BEACH	VA	BUICK GMC	SOUTHERN BUICK GMC LYNNHHA	2375 VIRGINIA BEACH BLVD	VIRGINIA BEAC	VA	23454	(757) 340-0800	<a href="https://www.southernbuickgmclynnhaha.com">https://www.southernbuickgmclynnhaha.com</a>	
SUBARU	5	VA BEACH	VA	BUICK GMC	SUTTLE MOTOR CORPORATION	12525 JEFFERSON AVE	NEWPORT NEW	VA	23602	(757) 886-1700	<a href="https://www.suttlemotorcorporation.com">https://www.suttlemotorcorporation.com</a>	
VW	4	VA BEACH	VA	BUICK GMC	SUTTLE MOTOR CORPORATION	12525 JEFFERSON AVE	NEWPORT NEW	VA	23602	(757) 424-6380	<a href="https://www.suttlemotorcorporation.com">https://www.suttlemotorcorporation.com</a>	
LEXUS	2	VA BEACH	VA	BUICK GMC	S CADILLAC, BUICK, GMC TRUCKS	1197 US HIGHWAY 17 S	CHESAPEAKE	VA	23320	(757) 424-6380	<a href="https://www.scadillac.com">https://www.scadillac.com</a>	
CADILLAC	3	VA BEACH	VA	CADILLAC	S CADILLAC, BUICK, GMC TRUCKS	1197 US HIGHWAY 17 S	ELIZABETH CITY	VA	27909	(252) 338-2131	<a href="https://www.scadillac.com">https://www.scadillac.com</a>	
ACURA	3	VA BEACH	VA	CADILLAC	SUTTLE MOTOR CORPORATION	12525 JEFFERSON AVE	NEWPORT NEW	VA	23602	(757) 349-6067	<a href="https://www.suttlemotorcorporation.com">https://www.suttlemotorcorporation.com</a>	
MERCEDES-BENZ	2	VA BEACH	VA	CADILLAC	RICK HENDRICK CADILLAC	6222 VIRGINIA BEACH BLVD	NORFOLK	VA	23502	(757) 401-4355	<a href="https://www.rickhendrickcadillac.com">https://www.rickhendrickcadillac.com</a>	
VOLVO	2	VA BEACH	VA	CADR JEEP	DUKE AUTOMOTIVE CORP.	2016 N MAIN ST	SUFFOLK	VA	23434	(757) 802-4268	<a href="https://www.dukeautomotive.com">https://www.dukeautomotive.com</a>	
INFINITI	1	VA BEACH	VA	CADR JEEP	Southern Chrysler Jeep At Greenbriar	1414 S Military Hwy	Chesapeake	VA	23320	(757) 424-4600	<a href="https://www.southernchryslerjeepatgreenbriar.com">https://www.southernchryslerjeepatgreenbriar.com</a>	
AUDI	2	VA BEACH	VA	CADR JEEP	Hall Chrysler Dodge Jeep Ram C	3353 Western Branch Blvd	Chesapeake	VA	23321	(757) 233-8200	<a href="https://www.hallchryslerdodgejeepram.com">https://www.hallchryslerdodgejeepram.com</a>	
LINCOLN	2	VA BEACH	VA	CADR JEEP	Starz Motors Incorporated	2584 Pruden Blvd	Suffolk	VA	23434	(757) 539-0214	<a href="https://www.starzmotorsinc.com">https://www.starzmotorsinc.com</a>	
BMW	2	VA BEACH	VA	CADR JEEP	Hall Chrysler Dodge Jeep Ram V	3152 Virginia Beach Blvd	Virginia Beach	VA	23452	(757) 932-6501	<a href="https://www.hallchryslerdodgejeepramv.com">https://www.hallchryslerdodgejeepramv.com</a>	
TOTAL	90	VA BEACH	VA	CADR JEEP	Southern Dodge Chrysler Jeep R	2747 N Military Hwy	Norfolk	VA	23518	(757) 855-2277	<a href="https://www.southerndodgechryslerjeep.com">https://www.southerndodgechryslerjeep.com</a>	
		VA BEACH	VA	CADR JEEP	Pomoco Chrysler Jeep Dodge	12629 Jefferson Ave	Newport News	VA	23602	(757) 833-8001	<a href="https://www.pomocochryslerjeepdodge.com">https://www.pomocochryslerjeepdodge.com</a>	
		VA BEACH	VA	CADR JEEP	Pomoco Chrysler Dodge Jeep Ram	4116 W Mercury Blvd	Hampton	VA	23666	(757) 826-1100	<a href="https://www.pomocochryslerdodgejeepram.com">https://www.pomocochryslerdodgejeepram.com</a>	
		VA BEACH	VA	CADR JEEP	Crossroads Chrysler Jeep Dodge	4510 Whitehill Blvd	Prince George	VA	23875	(804) 733-4664	<a href="https://www.crossroadschryslerjeepdodge.com">https://www.crossroadschryslerjeepdodge.com</a>	
		VA BEACH	VA	CHEVROLET	R. K. CHEVROLET, INC.	2661 VIRGINIA BEACH BLVD	VIRGINIA BEAC	VA	23452	(757) 486-2222	<a href="https://www.rkchevrolet.com">https://www.rkchevrolet.com</a>	
		VA BEACH	VA	CHEVROLET	RICK HENDRICK CHEVROLET	6252 VIRGINIA BEACH BLVD NORFOLK, VA 2	VA	23502	(757) 455-4500			

	N	O	P	Q	R	S	T	U	V	W	X	Y
1	Google Review Score	Number of Google Reviews	YELP Review Scores	Number of Yelp Reviews	Cars.com Review Scores	Number of Cars.com Reviews	DealerRater Review Scores	Number of DealerRater Reviews	FaceBook Review Scores	Number of Facebook Reviews	BBB Rating	BBB.org Number of Complaints (3 Year)
2												
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												
26												

Figure 5.1.2 (a) CSV file Input Data

## ○ Designing Python Scrapper for all the websites to get reviews and rating

```

for i in range(0,df_cities["Range"][c]):
    google=str(df["Google Review Score"][i])
    if google == "nan":
        try:
            params = {
                "q": df["Dealerships Name"][i] + " " + df["Address"][i],
                "hl": "en",
                "gl": "us",
                "google_domain": "google.com",
                "api_key": "cdf70a41c71298320361b1a743622db220a213cc4878fc16b1032d36d38510f2"
            }
            client = GoogleSearchResults(params)
            results = client.get_dict()
            print(df["Dealerships Name"][i] + " " + df["Address"][i])
            for j in range(0,3):
                try:
                    df["Google Review Score"][i]=results['knowledge_graph']['rating']
                    df["Number of Google Reviews"][i]=results['knowledge_graph']['review_count']
                    print("Rating and Review of Google")
                    print(df["Google Review Score"][i],df["Number of Google Reviews"][i])
                    break
                except:
                    df["Google Review Score"][i]=""
                    df["Number of Google Reviews"][i]=""
            except:
                print("error in something")
                df["Google Review Score"][i]=""
                df["Number of Google Reviews"][i]=""
        except:
            print("error in something")
            df["Google Review Score"][i]=""
            df["Number of Google Reviews"][i]=""
```

**Figure 5.1.2 (b) Scrapper for Google Ratings and Review**

```

check=soup.findAll("div",{"class":"MuiPaper-root MuiCard-root styles__Rollup-y1p2dl-0 dgAmri MuiPaper-elevation1 MuiPaper-rounded"})
if len(check) == 0:
    review=soup.findAll("div", {"class": "MuiPaper-root MuiCard-root Card-sc-14y8sbi-0 TallCard-sc-1017v0f-0 fMyzRZ MuiPaper-elevation1 MuiPaper-rounded"})
    rating = soup.findAll("div", {"class": "MuiPaper-root MuiCard-root Card-sc-14y8sbi-0 elDRbn dtm-complaint-list MuiPaper-elevation1 MuiPaper-rounded"})
    try:
        df["BBB.org Number of Complaints (3 Year)"][i]=rating[0].find("p", {"class": "MuiTypography-root MuiTypography-body1"}).text
        if review[1].find("span", {"class": "LetterGrade-sc-1exw0et-0 bQtLRW"}) == None:
            tags = review[1].find("span", {"class": "LetterGrade-sc-1exw0et-0 bQtLRW dtm-rating"}).text
        else:
            tags = review[1].find("span", {"class": "LetterGrade-sc-1exw0et-0 bQtLRW"}).text
        df["BBB Rating"][i]=tags
        print(df["BBB Rating"][i],df["BBB.org Number of Complaints (3 Year)"][i])
    except:
        df["BBB Rating"][i]=""
        df["BBB.org Number of Complaints (3 Year)"][i]=""
        print("Not Available Reviews and Complaints")
    else:
        review = soup.findAll("div", {"class": "MuiGrid-root MuiGrid-container MuiGrid-spacing-xs-1 MuiGrid-align-items-s-xs-center"})
        rating = soup.findAll("div", {"class": "MuiPaper-root MuiCard-root styles__Rollup-y1p2dl-0 dgAmri MuiPaper-elevation1 MuiPaper-rounded"})
        try:
            df["BBB Rating"][i]=rating[0].div.span.text
            df["BBB.org Number of Complaints (3 Year)"][i]=review[0].div.span.text
            print(df["BBB Rating"][i],df["BBB.org Number of Complaints (3 Year)"][i])
        except:
            print("Not Available Reviews and Complaints")
            df["BBB Rating"][i]=""
            df["BBB.org Number of Complaints (3 Year)"][i]=""
    except:
        print("error in something")
        df["BBB Rating"][[i]]=""
```

**Figure 5.1.2 (c) Scrapper for bbb.org Ratings and Review**

```

from serpapi.google_search_results import GoogleSearchResults
df["Cars.com Review Scores"].fillna("No Data", inplace = True)
for i in range(0,df_cities["Range"][c]):
    if df["Cars.com Review Scores"][i] == "No Data":
        count=count+1
    try:
        params = {
            "q": df["Dealerships Name"][i] + " " + df["City"][i] + " Cars.com",
            "hl": "en",
            "gl": "us",
            "google_domain": "google.com",
            "api_key": "cdf70a41c71298320361b1a743622db220a213cc4878fc16b1032d36d38510f2"
        }
        client = GoogleSearchResults(params)
        results = client.get_dict()
        print("Cars.com")
        print(df["Dealerships Name"][i] + " " + df["City"][i] + " Cars.com")
        for j in range(0,3):
            try:
                df["Cars.com Review Scores"][i]=results['organic_results'][j]['rich_snippet']['top']['detected_extensions']['rating']
                df["Number of Cars.com Reviews"][i]=results['organic_results'][j]['rich_snippet']['top']['detected_extensions']['reviews']
                print(df["Dealerships Name"][i])
                print("Rating and Review of Dealer Rater")
                print(df["Cars.com Review Scores"][i],df["Number of Cars.com Reviews"][i])
                break
            except:
                df["Cars.com Review Scores"][i]=""
                df["Number of Cars.com Reviews"][i]=""
        except:
            print("error in something")
            df["Cars.com Review Scores"][i]=""
            df["Number of Cars.com Reviews"][i]="""
    
```

**Figure 5.1.2 (d) Scrapper for cars.com Ratings and Review**

```

for i in range(0,df_cities["Range"][c]):
    dealer_rater=str(df["Dealerrater Review Scores"][i])
    if dealer_rater == "nan":
        try:
            params = {
                "q": df["Dealerships Name"][i] + df["City"][i] + "Dealer Rater",
                "hl": "en",
                "gl": "us",
                "google_domain": "google.com",
                "api_key": "Key"
            }
            client = GoogleSearchResults(params)
            results = client.get_dict()

            for j in range(0,4):
                try:
                    df["Dealerrater Review Scores"][i]=results['organic_results'][j]['rich_snippet']['top']['detected_extensions']['rating']
                    df["Number of Dealerrater Reviews"][i]=results['organic_results'][j]['rich_snippet']['top']['detected_extensions']['reviews']
                    print(df["Dealerships Name"][i])
                    print("Rating and Review of Dealer Rater")
                    print(df["Dealerrater Review Scores"][i],df["Number of Dealerrater Reviews"][i])
                    break
                except:
                    df["Dealerrater Review Scores"][i]=""
                    df["Number of Dealerrater Reviews"][i]=""
            except:
                print("error in something")
                df["Dealerrater Review Scores"][i]=""
                df["Number of Dealerrater Reviews"][i]="""
    
```

**Figure 5.1.2 (e) Scrapper for Dealer -Rater Ratings and Review**

```

from serpapi.google_search_results import GoogleSearchResults
df["FaceBook Review Scores"].fillna("No Data", inplace = True)
for i in range(0,df_cities["Range"])[c]:
    if df["FaceBook Review Scores"][i] == "No Data":
        count=count+1
    try:
        print("Facebook")
        url ="https://serpapi.com/search?engine=yahoo&p=" + df["Dealerships Name"][i] + " "+df["City"][i] + "Facebook" +
"&api_key=0b0f0784f2ab0c290db9d97a5dd6bd8903fbba06a478bcb2eb14d4fe0be9daf0"
        response = requests.request('GET', url)
        e=response.json()
        for j in range(0,5):
            try:
                df["FaceBook Review Scores"][i]=e['organic_results'][j]['rich_snippet']['rating']
                df["Number of Facebook Reviews"][i]=e['organic_results'][j]['rich_snippet']['reviews']
                print(df["Dealerships Name"][i])
                print("Rating and Review of Facebook")
                print(df["FaceBook Review Scores"][i],df["Number of Facebook Reviews"][i])
                break
            except:
                df["FaceBook Review Scores"][i]=""
                df["Number of Facebook Reviews"][i] ""
        except:
            print("error in something")
            df["FaceBook Review Scores"][i]=""
            df["Number of Facebook Reviews"][i] ""
    except:
        print("error in something")
        df["FaceBook Review Scores"][i]=""
        df["Number of Facebook Reviews"][i] ""

```

**Figure 5.1.2 (f) Scrapper for Facebook Ratings and Review**

```

url = 'https://api.yelp.com/v3/businesses/search'
headers = {'Authorization': 'Bearer {}'.format(API_KEY)}

url_params = {
    'term': DEFAULT_TERM,
    'location': DEFAULT_LOCATION,
    'limit': SEARCH_LIMIT
}
response = requests.request('GET', url, headers=headers, params=url_params)

for business in response.json()["businesses"]:
    if len(business)==0:
        df["YELP Review Scores"][i] = "No Data"
        df["Number of Yelp Reviews"][i] = "No Data"
    else:
        df["Number of Yelp Reviews"][i]=business["review_count"]
        df["YELP Review Scores"][i]=business["rating"]
        print(business["review_count"])
        print(business["rating"])

print()

```

**Figure 5.1.2 (g) Scrapper for YELP Ratings and Review**

## ○ Final output CSV file with scrapped data

City	State	Postcode	Phone Number	Website	Google Review Score	Number of Google Reviews	Number of Yelp Reviews	YELP Review Scores	Cars.com Review Scores	Number of Cars.com	
San Antonio	TX	78230	877-934-5933	https://www.gunnacura.com/	4.7	1668	23	4.5	4.8	659	
NORTH SELMA	TX	78154	(844) 846-6947		4.4	274	135	4.5	4.8	5	
San Antonio		78257	(888) 710-3466		4.3	837	586	4	4.2	300	
San Antonio	TX	78249	(210) 732-7121		4.4	2105	218	4	4.5	506	
BOERNE	TX	78006	(830) 981-9000	https://www.ancirabuickgmc.com/	4.8	431	188	4.5	4.9	55	
PLEASANTON	TX	78064	(830) 569-2241	https://www.purschmotors.net/	4.3	95	26	4	5	22	
NORTH SELMA	TX	78154	(210) 599-5600	https://www.gunnabuickgmc.com/	4.5	1550	19	5	4.6	203	
SEGUIN	TX	78155	(830) 303-4546	https://www.soechtingmotors.net/	4.6	87	36	4.5			
SAN ANTONIO	TX	78252	(210) 490-2000	https://www.cavenderbuickgmc281.com/	4.7	1776	1	5	4.5	39	
SAN ANTONIO	TX	78254	(210) 819-4444	https://www.cavenderbuickgmcwest.com/	4.7	2088	21	5	4.5	35	
SAN ANTONIO	TX	78230	(210) 263-3657	https://www.kenbatchelcorcadillac.com/	4.5	521	89	4.5	3.9	37	
SAN ANTONIO	TX	78233	(210) 265-6098	https://www.cavendercadillac.com/	4.7	535	162	3.5	4.5	38	
Boerne	TX	78006	(830) 755-8000	http://www.boernedodgechryslerjeep.com/	4.3	1288	188	4.5	4.2	70	
Castroville	TX	78009	(210) 890-6989	https://www.nytemaxwellcastroville.com/	4.9	147	38	4.5			
Devine	TX	78016	(830) 665-6401	https://www.brownododgechryslerjeep.com/	4.1	147	21	5	5	302	
				https://www.allwaysautogroup.com/							
Pleasanton	TX	78064	(830) 281-2244		4	202	26	4			
Floresville	TX	78114	(830) 216-4216	https://www.pricechryslerdodgejeep.com/	4.9	170	4.9	26	3.9	8	
				https://www.bbmotors.com/							
Braunfels	TX	78130	(830) 606-8500		4.4	23	32	4.5	4.3	45	
Seguin	TX	78155	(830) 372-3444	http://www.yaklincdjr.com/	3.9	259	36	4.5	4.7	42	
San Antonio	TX	78211	(210) 828-1515	https://www.lonestarchryslerdodgejeepsanantonio.com	3.8	697	629	4	3.8	20	
San Antonio	TX	78216	(210) 249-7500	https://www.northstarodge.net/	3.8	1018	9	4	3.7	48	
San Antonio	TX	78230	(210) 558-1500	https://www.ancirajcd.com/	4.4	1983	356	4.5	4.8	112	
San Antonio	TX	78233	(210) 967-2000	https://www.sanantoniododgechryslerjeppram.com/			254	4.5	4.4	1437	
San Antonio	TX	78238	(210) 684-6610	https://www.ingramparkcj.net/	4.3	1943	27	3.5	4.8	1259	
Pearland	TX	77581	(281) 485-1495	http://chevrolet.com/	4.1	625	57	3.5	4.8	127	
Boerne	TX	78006	(210) 477-9300	http://www.cavenderchevrolet.com/Service	4.8	1117	188	4.5	4.3	26	
Devine	TX	78016	(830) 665-4435	http://www.brownchevrolet.com/	4.5	172	247	4.5	5	1	
Floresville	TX	78114	(830) 216-4216	http://www.chevydeal.com/			33	4	4.5	11	
La Vernia	TX	78121	(830) 253-8000	http://keepchevy.com/	4.2	133	90	3.5	4.7	27	
New Braunfels	TX	78130	(830) 606-3451	http://www.valmarkchevy.com/	4.1	19	376	4.5	4.3	29	
Seima	TX	78154	(210) 599-5000	http://www.gunnchevrolet.com/	4.6	2734	3	5	4.5	1163	
				https://www.bbmotors.com/							
Cars.com Review Scores					Number of Cars.com	DealerRate Review Score	Number of DealerRate Review	FaceBook Review Score	Number of Facebook Review	BBB Rating	BB.org Number of Complaints (3 Year)
4.8		659	4.8	653	3	64	A+			8	
4.8		5	4.8	315	4,4	33	F			7	
4.2		300	4.2	193	4,2	249				5	
4.5		506	4.6	394	4,3	450	A+			19	
4.9		55	3.5	20	4,8	42	A+			5	
5		22	4.9	12	4,6	127					
4.6		203	4.6	148	4,2	107	A+			0	
			3.5	5	4,8	13	A+			0	
4.5		39	2.7	68	4,8	230	A-			8	
4.5		35	1.2	4	4,7	254	B+			4	
3.9		37	3.9	37	4,6	195	C-			6	
4.5		38	5	77	4,5	106	A-			5	
4.2		70	4.5	467	4,4	317	F			9	
			5	4			A+			14	
5		302	3.9	1224	4,2	36	B-			16	
			4.7	479	4,2	53	A+			1	
3.9		8	4.9	31	4,9	26	NR			0	
4.3		45	4.4	110	4,2	198	A+			4	
4.7		42	2	39	4,3	41	A+			1	
3.8		20	4	28	3,2	46	A-			5	
3.7		48	3.1	20	3,8	30	A+			17	
4.8		112	4.5	552	4,4	225	A+			3	
4.4		1437	4.5	1102	2	158	F			33	
4.8		1259	4.8	1207	3,9	78	A+			25	
4.8		127	1	3951	4,4	236	NR			3	
4.3		26	3.5	10	4,3	122	A+			0	
5		1	4.8	445	4,6	66	A+			5	
4.5		11	4.7	192	4,1	51	A+			1	
4.7		27	4.7	121	4,7	121	A+			2	
4.3		29	5	373			A+			2	
4.5		1163	4.6	514	4,5	434	A+			5	
			4.6	510	4,5	746	A+				
N ANTONIO SFATLIE ST LOUIS TAMPA TUCSON VIRGINIA BEACH					(+)	:	4				

```

sum=0
import requests
import pandas as pd
for c in range(0,40):
    count=0
    df_cities=pd.read_csv("The file path with name cities")
    print(df_cities)
    df=df_cities[["City"]].copy()
    df["Field You want to check"] = df["Field You want to check"].fillna("No Data", inplace = True)
    for i in range(0,int(df["Range"])[c]):
        if df["Field You want to check"][i] == "No Data":
            count=count+1
    print("count:",count)
    sum=sum+count
print(sum)

```

Figure 5.1.2 (h) Final Output with Script to check missing values

- We have achieved **97% of data accuracy** and some data are missing because of issues in dealership names and some are not registered on the website or not registered with the same dealership name and we got **67412 data from 69612**.

## 5.2 Flutter Applications

### 5.2.1 Registration and Login Application with Flutter and Firebase

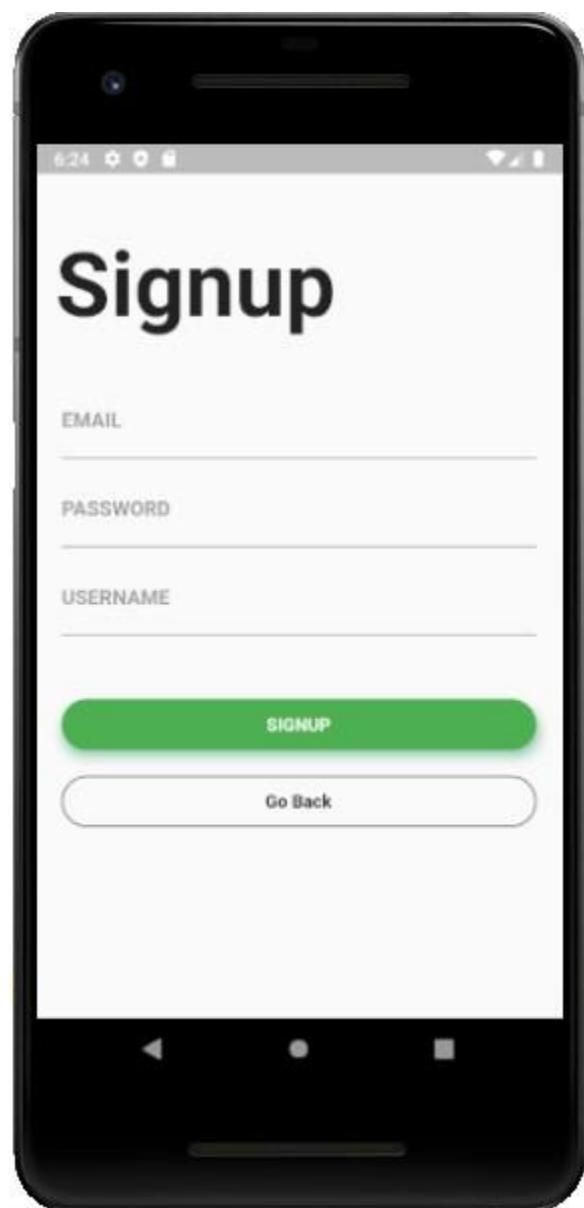


Figure 5.2.1 (a) Registration Form Flutter

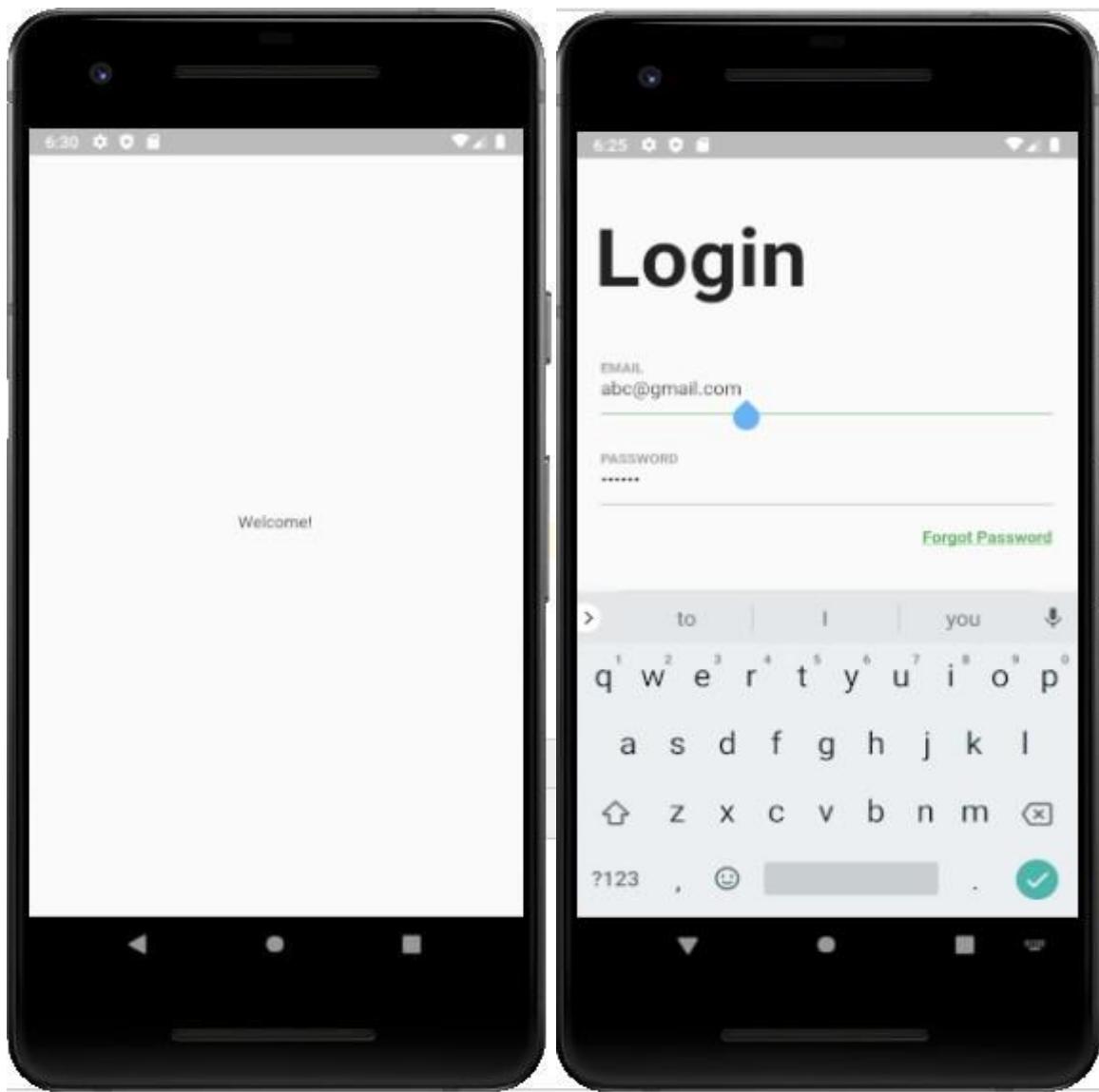


Figure 5.2.1 (b) Login Form Flutter

The image is a screenshot of the Firebase Database console. It shows a table of user data with columns for Identifier, Providers, Created, Signed In, and User UID. The table lists ten users, each associated with an email address and a unique User UID. The interface includes a search bar at the top and navigation controls at the bottom.

Identifier	Providers	Created	Signed In	User UID
harsh@gmail.com	✉	Apr 15, 2020	Apr 15, 2020	425Rk9jG4eessGjySoyez1wD9p1
k@q.com	✉	Apr 15, 2020	Apr 15, 2020	DhBBDNA5eUw/gAXTEmzb8ILB3
anu@gmail.com	✉	Apr 15, 2020	Apr 15, 2020	FCKIDZ58UJAanTyJomJPqSGC...
abc@gmail.com	✉	Apr 14, 2020	Apr 15, 2020	YlmrlM4QvUO3c27uIdP7MadM2
nick@yopmail.com	✉	Apr 15, 2020	Apr 15, 2020	a0tuqPamYVlNlwihlgDMBSiGSK1
khushali@gmail.com	✉	Apr 15, 2020	Apr 15, 2020	hlnldqgrpb8E4bTSFgylbPsd5s42
anuj@gmail.com	✉	Apr 15, 2020	Apr 15, 2020	nfvuPjyYMGq0bdy7nhb05aT2E3
hello@gmail.com	✉	Apr 15, 2020	Apr 15, 2020	omPtyaVRJXOD00xxRCXT5vRlx2
biz@gmail.com	✉	Apr 15, 2020	Apr 17, 2020	uxCeaOHOpfeORJAgzHbiQdk2

Figure 5.2.1 (c) Firebase Database

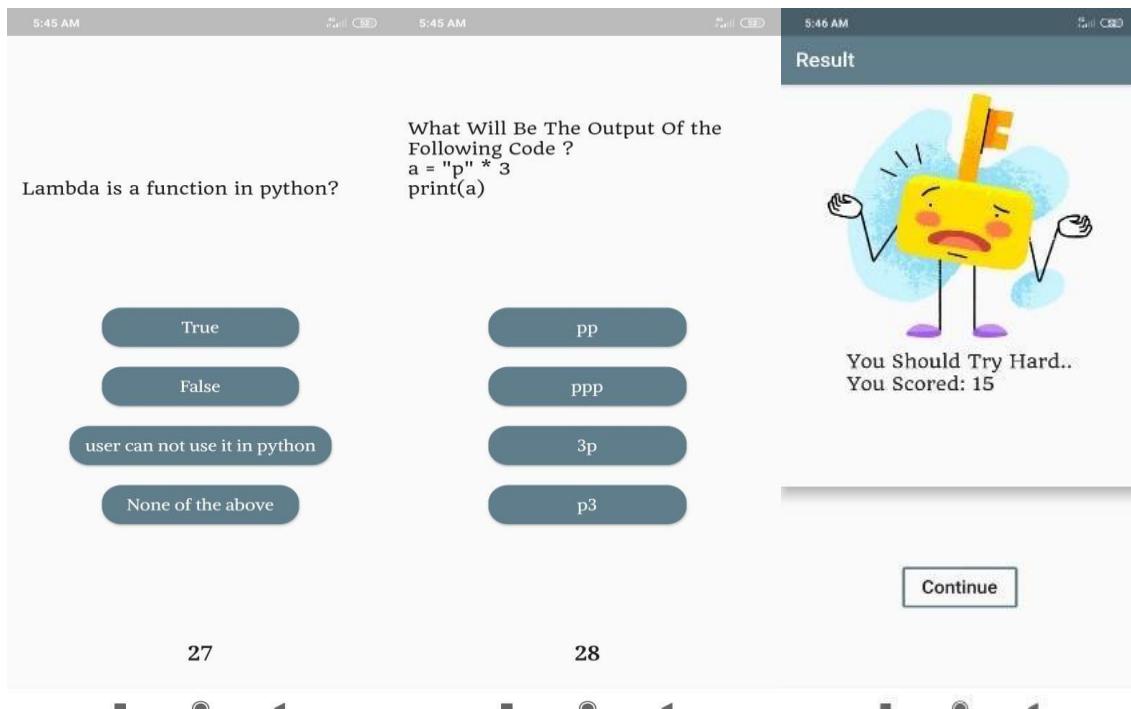
### 5.2.2 Quiz Application with Flutter

- Quiz App Home Page



Figure 5.2.2 (a) Quiz App Home Page

- Questions in various languages with Timer and Display Result



27

28

Figure 5.2.2 (b) Quiz App Testing

## 5.3 COVID-19 Patient Prediction System with Machine Learning

### 5.3.1 Module:1 Parameter selection and Final Input data

Age	Fever	Fatigue	Runny_Nose	Difficulty_in_breath	Infection_probability
23	101.2922402999999	0	1	0	1.0
2	98.71632562	1	0	0	0.0
43	99.47179211	0	0	1	0.0
48	98.33631064	0	1	0	0.0
70	101.9665676	1	0	0	1.0
52	100.1072293	1	0	0	1.0
30	99.51858222	1	1	0	0.0
97	98.07185085	1	1	0	0.0
28	98.95834761	1	1	0	0.0
12	101.1153794000001	1	0	0	1.0
71	101.6030552	1	1	0	1.0
82	98.12270966	0	0	0	0.0
49	100.388933	1	1	0	1.0
81	99.36682977	1	0	0	0.0
29	98.18366367	1	1	1	1.0
69	98.65370215	1	0	0	0.0
11	99.7427881999999	1	1	0	0.0
6	99.95122959	0	0	1	0.0

### 5.3.2 Module:2 Developing training model algorithm in Python

```

import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
import pickle

def data_split(data,ratio):
    shuffled= np.random.permutation(len(data))
    test_set_size=int(len(data)* ratio)
    test_indices=shuffled[:test_set_size]
    train_indices=shuffled[test_set_size:]
    return data.iloc[train_indices],data.iloc[test_indices]

if __name__ == "__main__":
    df=pd.read_csv("Symptoms_Data.csv")
    train,test=data_split(df,0.2)
    X_train=train[["Age","Fever","Fatigue","Runny_Nose","Difficulty_in_breath"]].to_numpy()
    X_test=test[["Age","Fever","Fatigue","Runny_Nose","Difficulty_in_breath"]].to_numpy()
    Y_train=train[["Infection_probability"]].to_numpy().reshape(4000,)
    Y_test=test[["Infection_probability"]].to_numpy().reshape(999,)

    clf = LogisticRegression()
    clf.fit(X_train,Y_train)
    # open a file, where you ant to store the data
    file = open('model.pkl', 'wb')

    # dump information to that file
    pickle.dump(clf, file)

```

### 5.3.3 Flask Web app development

The screenshot shows a web application titled "COVID-19 Detector". On the left side, there is a large blue gradient background with a white circular icon of a virus. Below the icon, two paragraphs of text provide information about the app's purpose and how it can help prioritize patients.

**COVID-19 Detector**

Enter age  Do you have Fatigue?

Enter Fever  Do you have runny nose?

Do you have difficulty in breathing?

**Submit**

Patient Status:  
Patient probability score:

Figure 5.3.3 Flask Webapp

### 5.3.4 Deployment of application and testing

This screenshot is identical to Figure 5.3.3, showing the "COVID-19 Detector" form. However, the "Patient Status" and "Patient probability score" fields now contain specific test results: "Consult Doctor as soon as possible! Don't Panic" and "Patient probability score: 70.1101440867" respectively.

**COVID-19 Detector**

Enter age  Do you have Fatigue?

Enter Fever  Do you have runny nose?

Do you have difficulty in breathing?

**Submit**

Patient Status: Consult Doctor as soon as possible! Don't Panic  
Patient probability score: 70.1101440867

Figure 5.3.4 (a) Test Results in app

The figure consists of two vertically stacked screenshots of a web application titled "COVID-19 Detector". Both screenshots have a blue header and footer with a white central content area. In the top-left corner of the content area, there is a small icon of a coronavirus.

**Screenshot 1 (Top):**

- Left Column:** Contains two text blocks:
  - "With this the person can self-asses and be sure and stay home."
  - "It can also help prioritise the patients who really have Coronavirus and help the health care workers who really have it and save kits for the needed."
- Right Column:** Contains a form with the following fields:
  - Input field: 21
  - Dropdown menu: No
  - Input field: 98
  - Dropdown menu: No
  - Input field: Yes
  - Submit button
- Bottom:** Displays "Patient Status:" and "Patient probability score:" followed by their respective values.

**Screenshot 2 (Bottom):**

- Left Column:** Contains two text blocks:
  - "With this the person can self-asses and be sure and stay home."
  - "It can also help prioritise the patients who really have Coronavirus and help the health care workers who really have it and save kits for the needed."
- Right Column:** Contains a form with the following fields:
  - Input field: Enter age
  - Dropdown menu: Do you have Fatigue?
  - Input field: Enter Fever
  - Dropdown menu: Do you have runny nose?
  - Input field: Do you have difficulty in breathing?
  - Submit button
- Bottom:** Displays "Patient Status: No Need To Worry or Consult Doctor Stay Home Stay Safe" and "Patient probability score: 46.412907817".

Figure 5.3.4 (b) Display results

**AWS EC2 for hosting the application:**

**Link of web app hosted:- <http://ec2-34-235-168-245.compute-1.amazonaws.com/>**

## 5.4 Django Hotel Management System Prototype for all Modules

### 5.4.1 Module-1 Registration Login with Twilio OTP

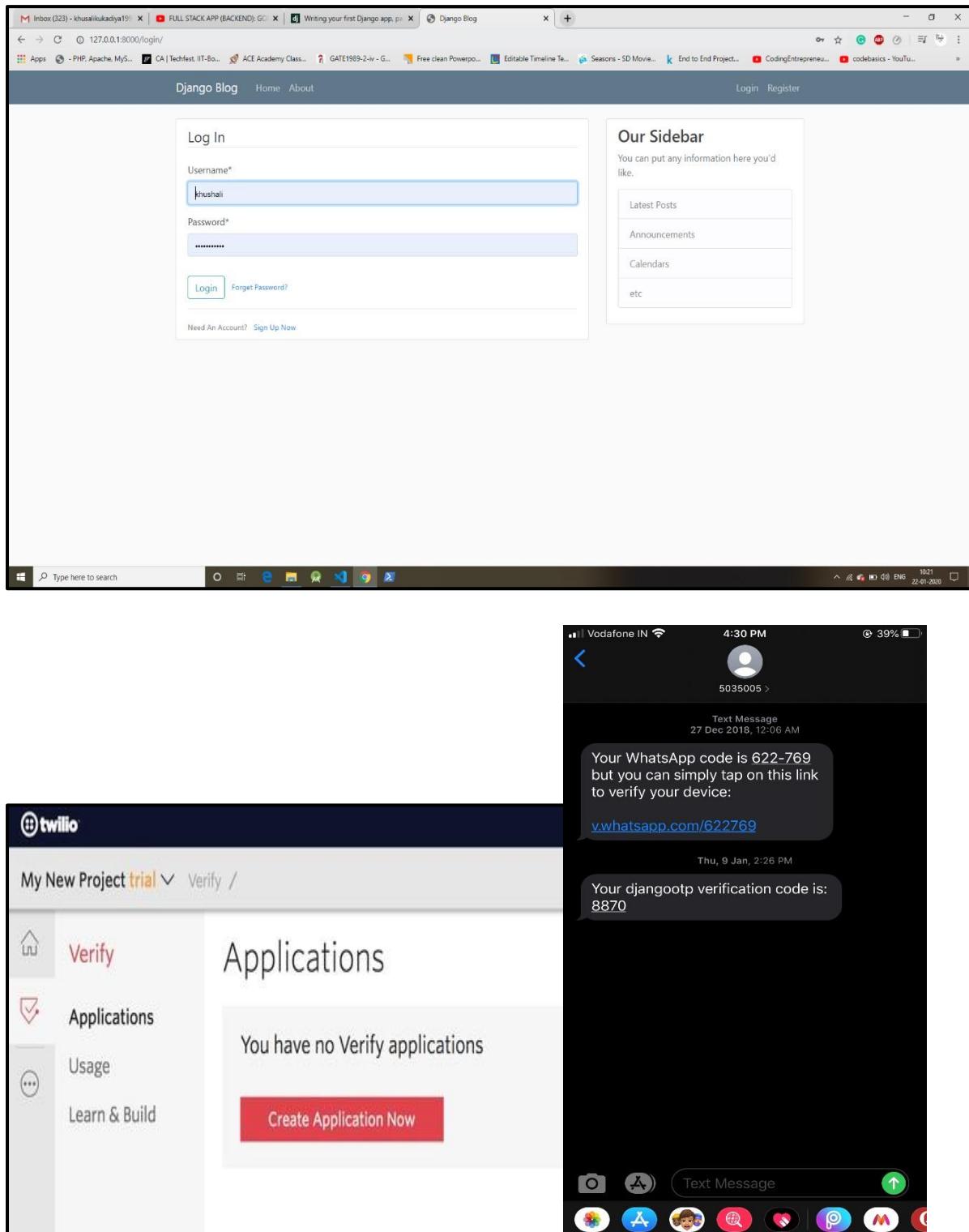
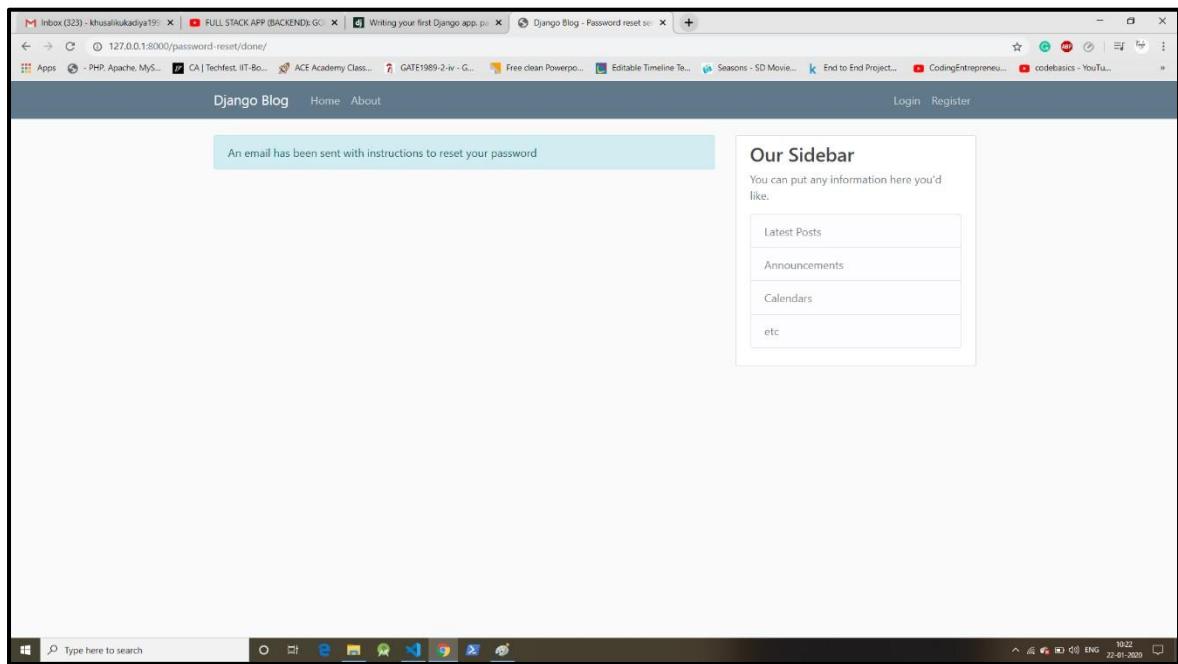
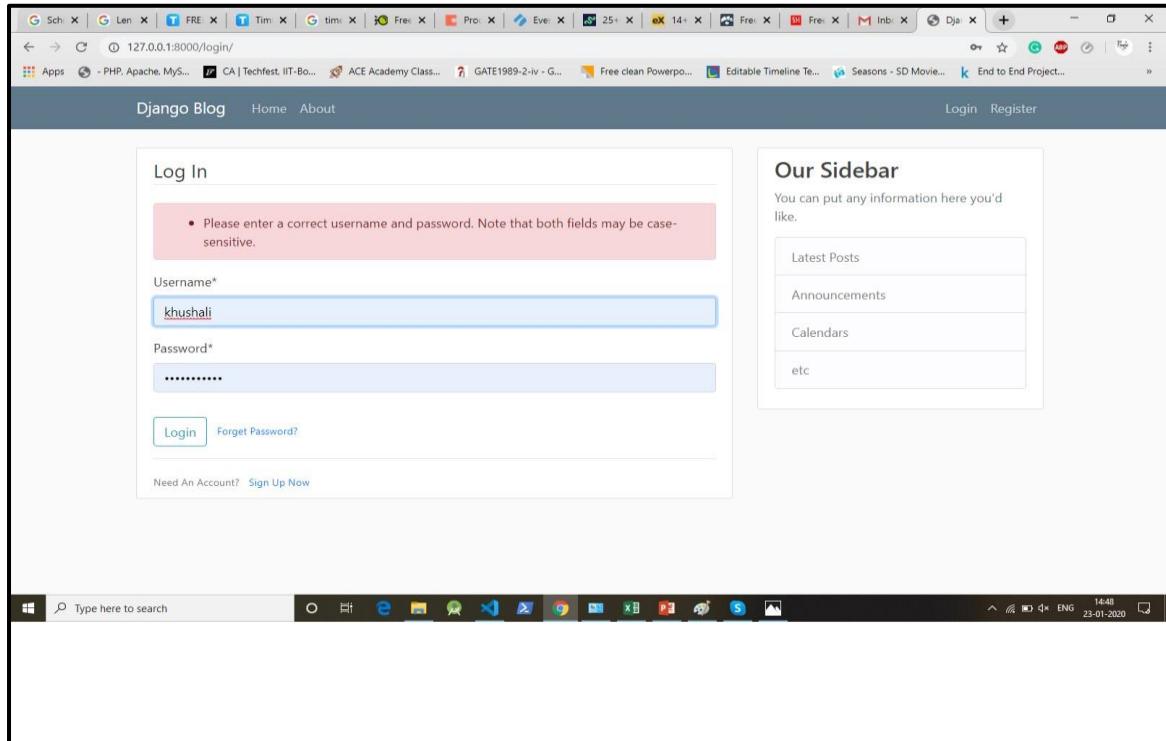


Figure 5.4.1 Django Register Login with OTP

### 5.4.2 Module:2 Email Password Reset



**Figure 5.4.2 (a) Login Form reset password link**

# Industrial Training Report

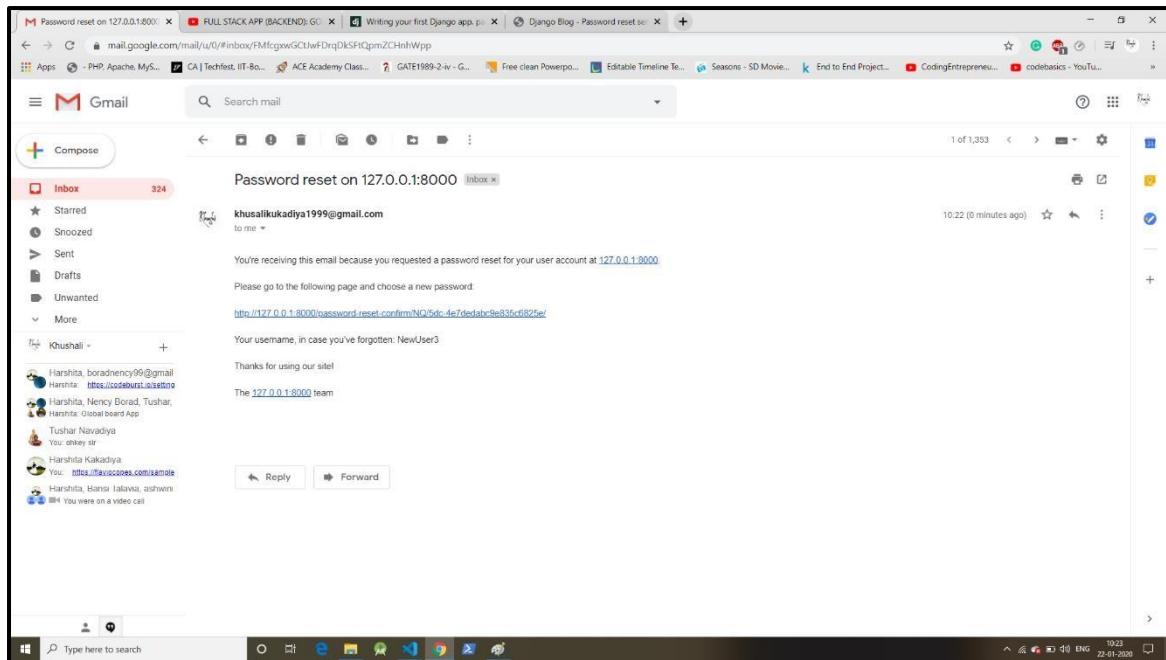


Figure 5.4.2 (b) An email password reset link

### 5.4.3 Payment Gateway with stripe

The image consists of two screenshots of a Django Stripe payment interface. The top screenshot shows a modal titled 'A Django Charge' with fields for email ('will@wsvincent.com'), card number ('4242 4242 4242 4242'), expiration ('01 / 20'), and CVC ('123'). A 'Remember me' checkbox is present. A large blue 'Pay \$5.00' button is at the bottom. A tooltip on the right side of the button states: 'You did not set a valid publishable key. Call Stripe.setPublishableKey() with your publishable key. For more info, see https://stripe.com/docs/stripe.js'. The bottom screenshot shows a 'Purchase' page with a green success message box containing the text 'Thanks for purchasing!'.

Figure 5.4.3 Stripe Payment Gateway

#### 5.4.4 ORM Postgre SQL Database

The screenshot shows the pgAdmin interface connected to a PostgreSQL database. The left sidebar displays the schema structure under the 'public' schema, including tables like auth\_group, auth\_permission, and auth\_user. The 'auth\_user' table is selected, and its data is displayed in a table below:

	<b>id</b>	<b>password</b>	<b>last_login</b>	<b>is_superuser</b>	<b>username</b>	<b>first_name</b>	<b>last_name</b>
1	1	pbkdf2_sha256\$180000\$doO...	2020-01-06 17:18:03.019508+05...	true	nency123		
2	2	pbkdf2_sha256\$180000\$FN...	2020-01-07 15:21:23.003237+05...	false	mansiip	mansi	patel
3	3	pbkdf2_sha256\$180000\$Lenl...	2020-01-07 15:47:52.296598+05...	false	hk	harshita	kakadiya
4	4	pbkdf2_sha256\$180000\$6gQ...	2020-01-07 15:48:39.817446+05...	false	kk	khushali	kukadiya

The right side of the interface features a 'Scratch Pad' window where a simple query is being typed:

```
1 SELECT * FROM public.auth_user
2
```

Below the scratch pad is another table view showing data from a different table:

<b>email</b>	<b>is_staff</b>	<b>is_active</b>	<b>date_joined</b>
boradnency99@gmail.com	true	true	2020-01-06 17:16:22.437796+05...
manhi@gmail.com	false	true	2020-01-07 12:37:58.828427+05...
harshi@gmail.com	false	true	2020-01-07 14:49:51.64699+05:30
kk@gmail.com	false	true	2020-01-07 15:23:21.970072+05...

Figure 5.4.4 Django ORM Database

### 5.4.5 CRUD Operations for Content Management

The figure consists of three vertically stacked screenshots of a Django-based Content Management System (HMS).

**Screenshot 1: Login Screen**

This screenshot shows the Django administration login page. The URL is 127.0.0.1:8000/admin/login/?next=/admin/. The page has a "Django administration" header and fields for "Username" (Harshita) and "Password". A "Log in" button is at the bottom.

**Screenshot 2: Product Creation Form**

This screenshot shows the "Add product" form under the "Products" section. The URL is 127.0.0.1:8000/admin/API/product/add/. The form fields include "Name" (HP Pavilion), "Description" (The HP Pavilion Gaming Laptop is designed so you can excel at any task. Gaming. Creation. Entertainment. A 9th Gen Intel Core processor and discrete graphics work together to jumpstart everything you do. Don't settle for average performance. Prepare yourself for a new level of power.), and "Price" (60000). Below the form are buttons for "Save and add another", "Save and continue editing", and a large blue "SAVE" button. A watermark for "Activate Windows" is visible at the bottom right.

**Screenshot 3: Success Confirmation**

This screenshot shows the confirmation message after a successful product creation. The URL is 127.0.0.1:8000/admin/API/product/add/. The message states: "The product "HP Pavilion" was added successfully. You may add another product below." Below this message is a link to "Add product".

Figure 5.4.5 (a) Create Operation in HMS

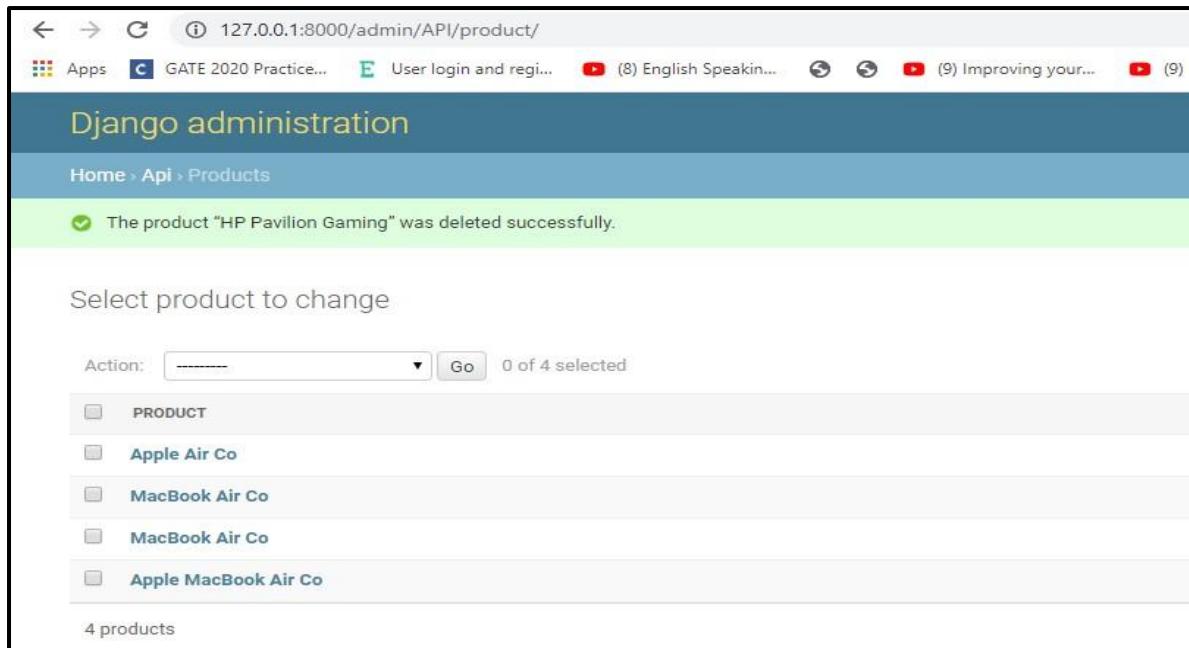


Figure 5.4.5 (b) Delete Operation in HMS

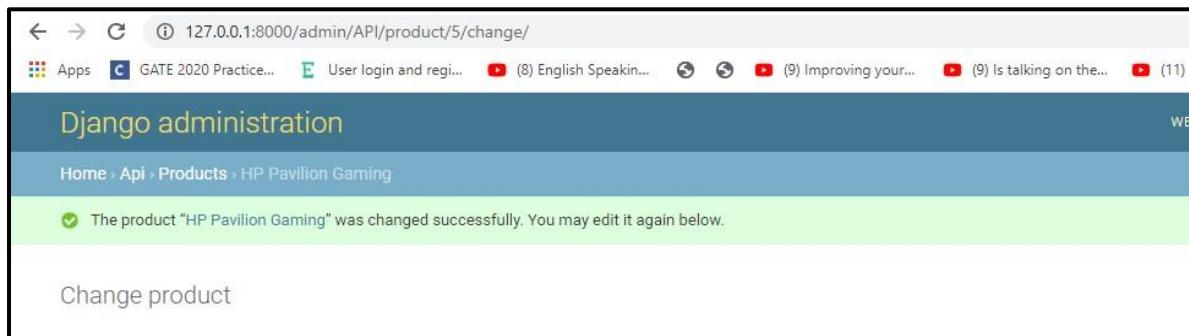


Figure 5.4.5 (c) Update Operation in HMS

Data Output Explain Messages Notifications						
	<b>id</b> [PK] integer	<b>created</b> timestamp with time zone	<b>name</b> character varying (20)	<b>description</b> text	<b>price</b> numeric (10,2)	
1	1	2020-01-04 13:00:35.505+05:30	Apple MacBook Air Co	It is fun to use, i...	61990.00	
2	2	2020-01-04 17:05:38.717+05:30	MacBook Air Co	It is fun to use, i...	65990.00	
3	3	2020-01-04 18:21:37.13+05:30	MacBook Air Co	It is fun to use, i...	65990.00	
4	4	2020-01-04 18:23:58.848+05:30	Apple Air Co	It is fun to use, i...	61990.00	
5	5	2020-01-23 17:57:25.971493+05:...	HP Pavilion Gaming	The HP Pavilion...	95000.00	

Figure 5.4.5 (d) Search Operation in HMS

## 5.5 AWS (Amazon Web Services) all task Implementation

### 5.5.1 Task-1:-Customer activity Tracker

- Flask application form for Customer for Input

The screenshot shows a web-based form for customer activity tracking. The fields are:

- customer profile number
- activity type
- product id
- account number
- campaign id
- device id
- email address
- phone number
- promocode
- reference id

Each field has a placeholder text (e.g., 'Enter customer profile number') and a green 'Submit' button at the bottom.

**Figure 5.5.1 (a) Flask app input form**

- AWS Lambda function in Python

```

lambda_function.py

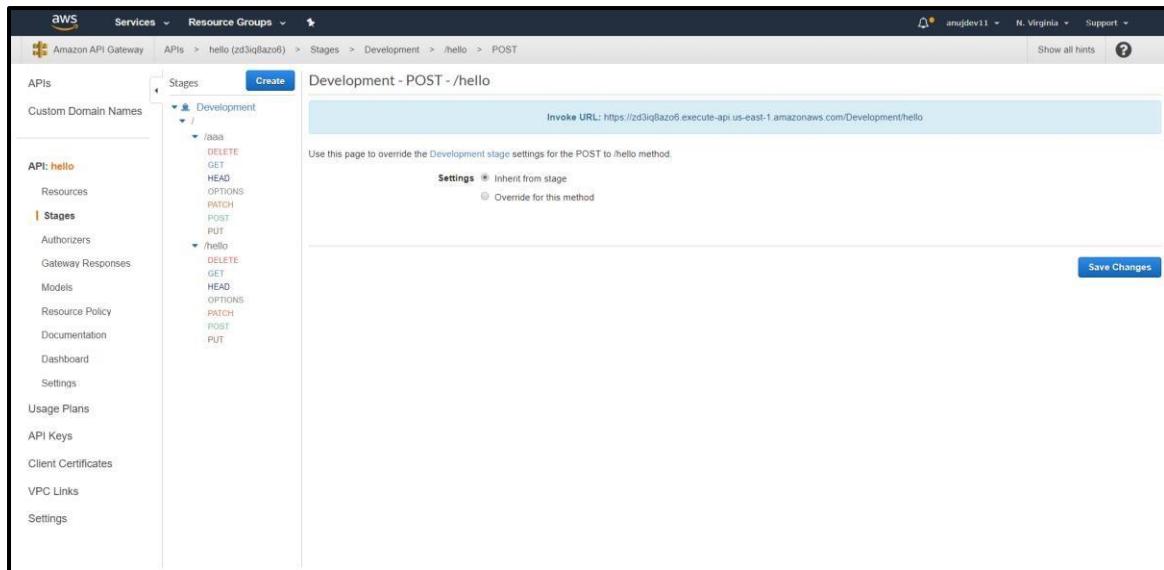
1 import boto3
2 import urllib
3 from datetime import datetime, date
4 import datetime
5 from botocore.errorfactory import ClientError
6 import botocore
7 s3_client = boto3.client("s3")
8
9 s3 = boto3.client("s3")
10 dynamodb = boto3.resource('dynamodb')
11 table = dynamodb.Table('try1')
12 def lambda_handler(event, context):
13     bucket_name_for_read=event['Records'][0]['s3']['bucket']['name']
14     s3_file_name=event['Records'][0]['s3']['object']['key']
15     resp=s3_client.get_object(Bucket=bucket_name_for_read, Key=s3_file_name)
16     data=resp['Body'].read().decode("utf-8")
17
18     bucket_name_for_write = "activitytrackentry"
19     today = str(date.today())
20     directory_name = today
21
22     ts = datetime.datetime.now().timestamp()
23     filename = "/" + str(ts) + ".csv"
24     s3.put_object(Bucket=bucket_name_for_write, Key=(directory_name+filename), Body=data)
25
26     data2=data.split("\n")
27     for x in data2:
28         print(x)
29         x_data=x.split(",")
30         #Adding the information to Dynamo Db
31         table.put_item(
32             Item={
33                 'customerprofilenumber':x_data[0],
34                 'activitytype':x_data[1],
35                 'activitydate':x_data[2],
36                 'productid':x_data[3],
37                 'accountnumber':x_data[4],
38                 'campaignid':x_data[5],
39                 'deviceid':x_data[6],
40                 'emailaddress':x_data[7],
41                 'insertdate':x_data[8],
42                 'lastupdatedate':x_data[9],
43                 'phonenumben':x_data[10],
44                 'promocode':x_data[11],
45                 'referenceid':x_data[12],
46             }
47         )

```

**Figure 5.5.1 (b) AWS Lambda Function**

- API Gateway with AWS Lambda

# Industrial Training Report



**Figure 5.5.1 (c) API Gateway**

## ○ Store Output in S3 Bucket

The screenshot shows the AWS S3 console for the "activitytrackerty" bucket. The "Overview" tab is selected. At the top, there are buttons for "Upload", "+ Create folder", "Download", and "Actions". The region is set to "US East (N. Virginia)". The main table lists objects in the bucket, all of which are folders named with dates from January 15 to January 22, 2020. The table includes columns for Name, Last modified, Size, and Storage class. A message at the bottom indicates "Viewing 1 to 5".

The screenshot shows the AWS S3 console for the "activitytrackerty" bucket, specifically viewing the "2020-01-22" folder. The "Overview" tab is selected. The main table lists two CSV files: "1579673056.733924.csv" and "1579673082.79316.csv". The table includes columns for Name, Last modified, Size, and Storage class. A message at the bottom indicates "Viewing 1 to 2".

- Store the Output in Dynamo DB

customerprofilei	accountnumber	activitydate	activitytype	campaignid	deviceid	emailaddress	insertdate	lastupdatedate	phonenumbers	productid	promocode
d	d	d	d	d	d	d	d	d	d	d	d
f	w	19:09:2	ww	w	w	19:09:2	19:09:2	ww	ww	w	w

Figure 5.5.1 (d) Output in S3 and Dynamo DB

### 5.5.2 Task-2: Customer Activity Tracker

- Dropped the CSV input file in the S3 bucket

Name	Last modified	Size	Storage class
1579590804.997925.csv	Jan 22, 2020 11:33:12 AM GMT+0530	75.0 B	Standard

Figure 5.5.2 (a) Input CSV file in S3

- Making AWS Lambda Trigger when a new file arrives in input in the S3 bucket

Name	Last modified	Size	Storage class
1579673056.733924.csv	Jan 22, 2020 11:34:17 AM GMT+0530	4.0 B	Standard
1579673082.79316.csv	Jan 22, 2020 11:34:43 AM GMT+0530	4.0 B	Standard

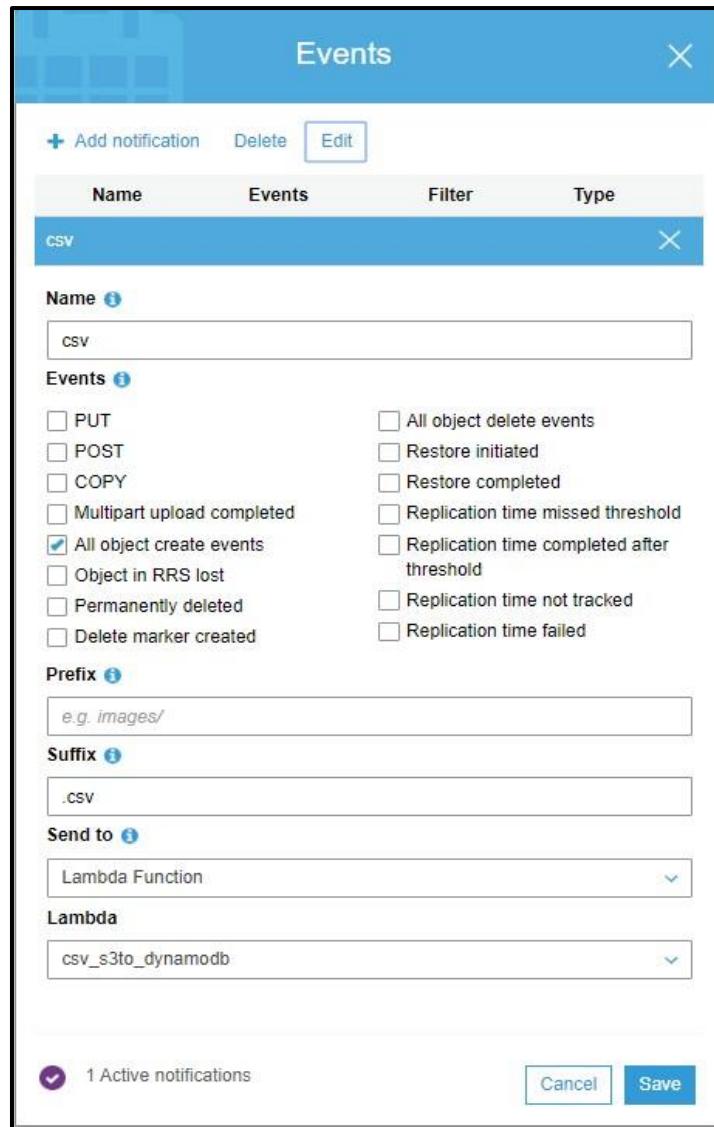


Figure 5.5.2 (b) AWS Lambda Trigger Event

### 5.5.3 Task-3 AWS Athena for Analysis

- Dropping CSV file as input in the S3 bucket

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	customerprofi	activitytype	activitydate	productid	accountnum	campaignid	deviceid	emailaddress	insertdate	lastupdatedate	phononenumber	promocode	referenceid
2	f	ww	22-12-2020	w	w	w	gj gj		22-12-2020	22-12-2020	ww	w	w
3	dshfg	hgfh	22-12-2020	jgjh	gjhg	jhg	jhg	jg	22-12-2020	22-12-2020	kj	hkh	kh
4	fdgfd	jhgj	22-12-2020	ghbj	JB bj	vj	vjh	jhj	22-12-2020	22-12-2020	j	vj	vj
5	HHHH	BJHGHGFH	22-12-2020	dfhgf	gfg	kjk	kjk	kb	22-12-2020	22-12-2020	jnk	nnk	knn
6	hj	gig	22-12-2020	gj	gjhg	ch	fh	gj gj	22-12-2020	22-12-2020	gj	gi	gj
7	jg	jg	22-12-2020	jgjh	JB bj	jgk	jhg	jk	22-12-2020	22-12-2020	gj	hj	hj
8	jg	jhgj	22-12-2020	kh	gjhg	gjhjhj	gj gj	jhf	22-12-2020	22-12-2020	hjg	jg	jgj
9													
10													

Figure 5.5.3 (a) Input CSV file in S3

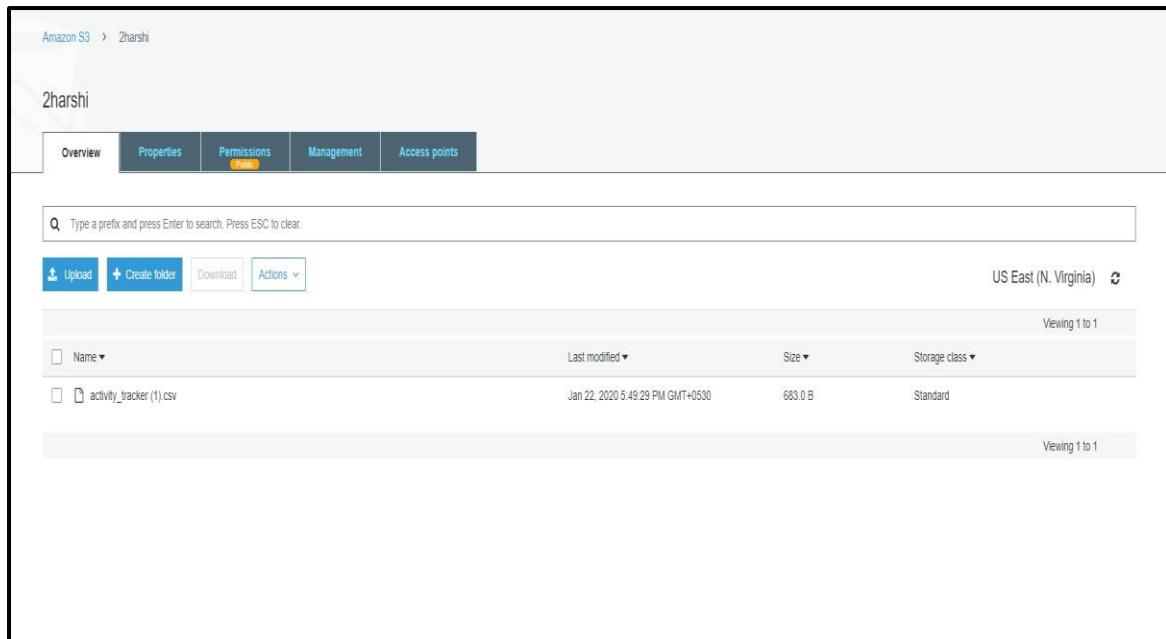


Figure 5.5.3 (b) Input CSV file in S3

- Writing Query in Athena to load the CSV file data

```

1 CREATE EXTERNAL TABLE IF NOT EXISTS elb_logs_raw_native_part (
2   request_timestamp string,
3   elb_name string,
4   request_ip string,
5   request_port int,
6   backend_ip string,
7   backend_port int,
8   request_processing_time double,
9   backend_processing_time double,
10  client_response_time double,
11  elb_response_code string,
12  backend_response_code string,
13  received_bytes bigint,
14  sent_bytes bigint,
15  request_verb string,
16  status string,
17  protocol string,
18  user_agent string,
19  ssl_cipher string,
20  ssl_protocol string )
21 PARTITIONED BY(year string, month string, day string)
22 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
23 WITH SERDEPROPERTIES (
24   'serialization.format' = '1', 'input.regex' = '([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^ ]*):([0-9]*) ([0-9]*) ([^.0-9]* ([.0-
25 LOCATION 's3://athena-examples/elb/raw/';

```

Run Query   Save As   Format Query   New Query ... (Run time: 2.03 seconds, Data scanned: 0KB)

Results

Query successful. If your table has partitions, you need to load these partitions to be able to query data. You can either [load all partitions](#) or [load them individually](#). If you use the `load all partitions (MSCK REPAIR TABLE)` command, partitions must be in a format understood by Hive. [Learn more](#).

Figure 5.5.3 (c) Athena Query

- Successfully data loaded in Athena

The screenshot shows a database interface with a sidebar on the left containing 'Data source' (awsdatasatalog), 'Database' (default), 'Tables (3)' (act241, act242, carts), and 'Views (0)'. The main area displays a query editor with the following SQL:

```
1: SELECT * FROM act242
```

Below the query editor, it says '(Run time: 1.7 seconds, Data scanned: 0.67 KB)'. The 'Run query' button is highlighted. The results section shows a table with 7 rows and columns including customerprofilenumber, activitiytype, activitiydate, productid, accountnumber, campaignid, deviceid, emailedaddress, insertdate, lastupdatedate, phononenumber, promocode, and refer.

	customerprofilenumber	activitiytype	activitiydate	productid	accountnumber	campaignid	deviceid	emailedaddress	insertdate	lastupdatedate	phononenumber	promocode	refer
1	vv	22-12-2020	w	w	gjgjgj			22-12-2020	22-12-2020	vv	v	w	
2	dshfgf	hgfh	22-12-2020	jgh	ghg	jhg	jhg	jg	22-12-2020	22-12-2020	kj	hkh	kh
3	fdfdf	jhgj	22-12-2020	gbhj	JBbjJ	vj	vjh	jhj	22-12-2020	22-12-2020	j	vj	vj
4	HHHHH	BJHGHGFGH	22-12-2020	dhgf	fg	jk	jk	kb	22-12-2020	22-12-2020	jnk	rnk	km
5	hj	gjg	22-12-2020	gl	ghg	ch	th	gjaiil	22-12-2020	22-12-2020	gi	gi	jij
6	jg	jg	22-12-2020	jgh	JBbjJ	jgk	jhg	jk	22-12-2020	22-12-2020	gi	hj	hj
7	jg	jgj	22-12-2020	kh	ghg	ghighi	gigig	jhf	22-12-2020	22-12-2020	hig	jg	jg

Figure 5.5.3 (d) Query Result

#### 5.5.4 Task-4 AWS Cloud Trail with AWS Lambda and S3 Bucket

- Creating the AWS cloud trail Logs and turn it on

The screenshot shows the 'Create Trail' configuration page. The 'Trails' section is selected. The 'Create Trail' form includes the following fields:

- Trail name\*: SampleTrail
- Apply trail to all regions: Yes (radio button selected)
- Create a new S3 bucket: Yes (radio button selected)
  - S3 bucket\*: samplebucket
  - Log file prefix: sampleprefix
  - Location: sampleprefix/AWSLogs/111111111111/CloudTrail/us-west-2
- Encrypt log files: No (radio button selected)
- Enable log file validation: Yes (radio button selected)
- Send SNS notification for every log file delivery: Yes (radio button selected)
  - Create a new SNS topic: sampleSNS

At the bottom, there is a note: '\* Required field' and 'Additional charges may apply'. The 'Create' button is highlighted.

Figure 5.5.4 (a) Creating AWS Cloud Trail

The screenshot shows the AWS CloudTrail Dashboard. On the left, there's a sidebar with links for CloudTrail, Dashboard, Event history, Trails, Learn more (Pricing, Documentation, Forums, FAQs), and View trails. The main area is titled "Recent events" and displays a table of five log entries. The columns are Event time, User name, Event name, and Resource type. The data is as follows:

Event time	User name	Event name	Resource type
2019-05-31, 12:32:09 PM	Mary_Major	ConsoleLogin	
2019-05-23, 03:13:27 PM	Mary_Major	DeleteBucket	S3 Bucket
2019-05-23, 03:13:27 PM	Mary_Major	DeleteBucket	S3 Bucket
2019-05-23, 03:13:25 PM	Mary_Major	DeleteRolePolicy	IAM Policy and 1 more
2019-05-23, 03:13:25 PM	Mary_Major	DeleteRole	IAM Role

At the bottom of the table, there's a link "View all events".

**Figure 5.5.4 (b) View the Cloud Trails**

#### ○ Cloud Trail Logs in S3 Bucket

The screenshot shows the AWS S3 console with the path "Amazon S3 > AWSLogs > CloudTrail". The main area is titled "Overview" and shows a list of 16 objects in an S3 bucket. The objects are listed by region: ap-northeast-1, ap-northeast-2, ap-northeast-3, ap-south-1, ap-southeast-1, ap-southeast-2, ca-central-1, eu-central-1, eu-west-1, eu-west-2, eu-west-3, sa-east-1, us-east-1, us-east-2, us-west-1, and us-west-2. Each object is represented by a small icon and a folder name. The columns in the table are Name, Last modified, Size, and Storage class. The status for all objects is "--". At the bottom, it says "Viewing 1 to 16".

**Figure 5.5.4 (c) S3 Bucket with Trails**

#### ○ AWS Lambda Function to read File from S3 Bucket

```

import boto3
import gzip
import json
import pandas as pd
from datetime import datetime, date
import datetime
from pathlib import Path

def csv(event, context):
    global reduced_item
    def to_string(s):
        try:
            return str(s)
        except:
            return s.encode('utf-8')
    def reduce_item(key, value):
        global reduced_item
        if type(value) is list:
            i=0
            for sub_item in value:
                reduce_item(key+'_'+to_string(i), sub_item)
                i=i+1
        elif type(value) is dict:
            sub_keys = value.keys()
            for sub_key in sub_keys:
                reduce_item(key+'_'+to_string(sub_key), value[sub_key])
        else:
            reduced_item[to_string(key)] = to_string(value)

    s3 = boto3.resource("s3")
    s3_client=boto3.client('s3')

```

- Normalize the nested JSON file with Python

```

for object_summary in bucket.objects.filter(Prefix="AWSLogs/792091115303/CloudTrail/"):
    print(object_summary.key)
    try:
        obj = s3.Object("trailforlambda",object_summary.key)
        with gzip.GzipFile(fileobj=obj.get()["Body"]) as gzipfile:
            raw_data = gzipfile.read()
        # print(raw_data)
        raw_data = json.loads(raw_data)
        print(raw_data)
        node = "Records"
        try:
            data_to_be_processed = raw_data[node]
        except:
            data_to_be_processed = raw_data
        processed_data = []
        header = []
        for item in data_to_be_processed:
            reduced_item={}
            reduce_item(node, item)
            header += reduced_item.keys()
            processed_data.append(reduced_item)
        header = list(set(header))
        header.sort()
        dfObj = pd.DataFrame(processed_data)
        # filename = Path(object_summary.key).stem
        filename = object_summary.key
        filename=filename.split(".")
        print(filename[0])
        dfObj.to_parquet('s3://cloudtrailresultfromlambda/' + filename[0] +'.parquet' ,engine='pyarrow')
        print ("Just completed writing csv file with %d columns" % len(header))
    except:
        pass

```

- Storing the Output File in Parquet compressed format in S3 Bucket in the date wise folder in the respected regions. Nested Folder Structure in AWS S3 Bucket as:

**AWSLogs** **user\_id\_folder** **CloudTrails** **regions\_folder** **date\_folder** **file**

The figure consists of three vertically stacked screenshots of the AWS S3 console interface.

**Screenshot 1:** Shows the root folder "cloudtrailtry12". The "AWSLogs" folder is visible under it. The interface includes standard S3 navigation and search tools.

Name	Last modified	Size	Storage class
AWSLogs	--	--	--

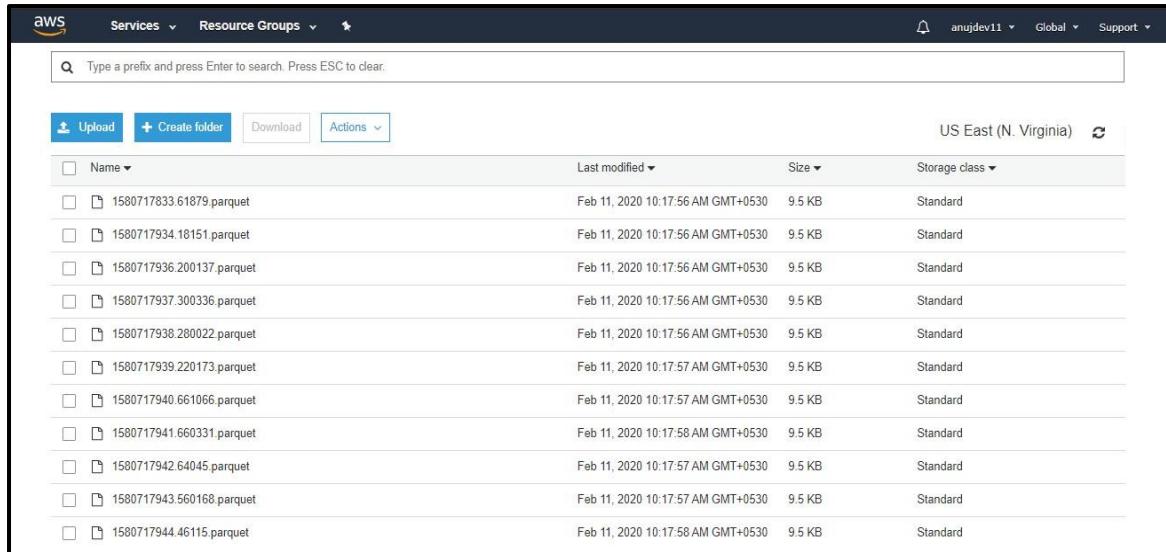
**Screenshot 2:** Shows the "AWSLogs" folder within "cloudtrailtry12". It contains three sub-folders: "CloudTrail-Digest", "CloudTrail-Insight", and "CloudTrail".

Name	Last modified	Size	Storage class
CloudTrail-Digest	--	--	--
CloudTrail-Insight	--	--	--
CloudTrail	--	--	--

**Screenshot 3:** Shows the "CloudTrail" folder from Screenshot 2. It contains four sub-folders: "dt=2020-02-02", "dt=2020-02-03", "dt=2020-02-10", and "dt=2020-02-11". The "dt=2020-02-02" folder is highlighted with a blue background.

Name	Last modified	Size	Storage class
dt=2020-02-02	--	--	--
dt=2020-02-03	--	--	--
dt=2020-02-10	--	--	--
dt=2020-02-11	--	--	--

**Figure 5.5.4 (d) Output nested folders in S3**



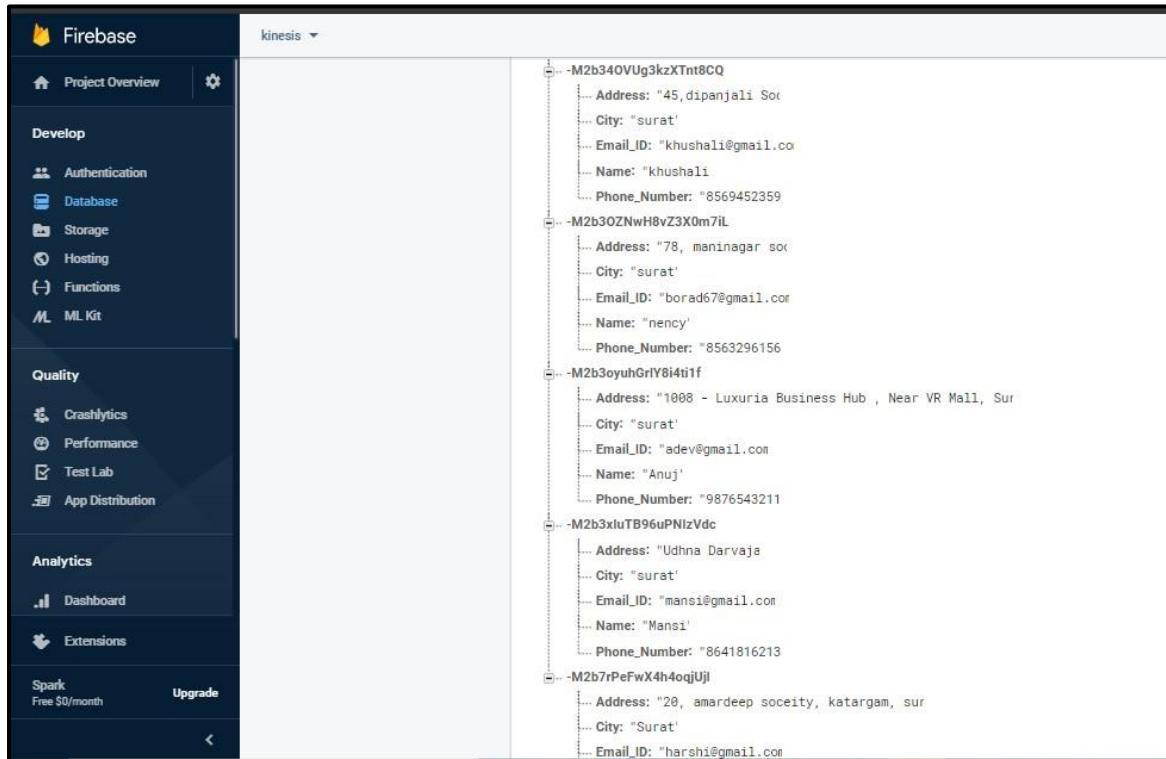
The screenshot shows the AWS S3 console interface. At the top, there are buttons for 'Upload', '+ Create folder', 'Download', and 'Actions'. A search bar is present with the placeholder 'Type a prefix and press Enter to search. Press ESC to clear.' To the right, it shows 'US East (N. Virginia)' and a refresh icon. Below the header is a table listing 12 parquet files. The columns are 'Name', 'Last modified', 'Size', and 'Storage class'. All files were last modified on Feb 11, 2020, at 10:17:56 AM GMT+0530, have a size of 9.5 KB, and are stored in the Standard storage class.

Name	Last modified	Size	Storage class
1580717833.61879.parquet	Feb 11, 2020 10:17:56 AM GMT+0530	9.5 KB	Standard
1580717934.18151.parquet	Feb 11, 2020 10:17:56 AM GMT+0530	9.5 KB	Standard
1580717936.200137.parquet	Feb 11, 2020 10:17:56 AM GMT+0530	9.5 KB	Standard
1580717937.300336.parquet	Feb 11, 2020 10:17:56 AM GMT+0530	9.5 KB	Standard
1580717938.280022.parquet	Feb 11, 2020 10:17:56 AM GMT+0530	9.5 KB	Standard
1580717939.220173.parquet	Feb 11, 2020 10:17:57 AM GMT+0530	9.5 KB	Standard
1580717940.661066.parquet	Feb 11, 2020 10:17:57 AM GMT+0530	9.5 KB	Standard
1580717941.660331.parquet	Feb 11, 2020 10:17:58 AM GMT+0530	9.5 KB	Standard
1580717942.64045.parquet	Feb 11, 2020 10:17:57 AM GMT+0530	9.5 KB	Standard
1580717943.560168.parquet	Feb 11, 2020 10:17:57 AM GMT+0530	9.5 KB	Standard
1580717944.46115.parquet	Feb 11, 2020 10:17:58 AM GMT+0530	9.5 KB	Standard

Figure 5.5.4 (e) Output parquet file in S3

### 5.5.5 AWS Kinesis with Firebase.

- Firebase Real-time Database to read the input data to kinesis



The screenshot shows the Firebase Real-time Database interface. On the left, there's a sidebar with 'Project Overview', 'Develop' (selected), 'Authentication', 'Database' (selected), 'Storage', 'Hosting', 'Functions', and 'ML Kit'. Under 'Quality', there are 'Crashlytics', 'Performance', 'Test Lab', and 'App Distribution'. Under 'Analytics', there are 'Dashboard' and 'Extensions'. At the bottom, it says 'Spark Free 50/month' and 'Upgrade'. The main area shows a tree view of database nodes. The root node has several child nodes, each containing user information: address, city, Email\_ID, Name, and Phone\_Number. The data is as follows:

- M2b340VUg3kzXTnt8CQ
  - Address: "45, dipanjali Soc"
  - City: "surat"
  - Email\_ID: "khushali@gmail.co"
  - Name: "khushali"
  - Phone\_Number: "8569452359"
- M2b30ZNwH8vZ3X0m7iL
  - Address: "78, maninagar soc"
  - City: "surat"
  - Email\_ID: "borad67@gmail.co"
  - Name: "nency"
  - Phone\_Number: "8563296156"
- M2b3oyuhGrIy8i4t1f
  - Address: "1008 - Luxuria Business Hub , Near VR Mall, Surat"
  - City: "surat"
  - Email\_ID: "adev@gmail.com"
  - Name: "Anuj"
  - Phone\_Number: "9876543211"
- M2b3xluTB96uPNizVdc
  - Address: "Udhna Darvaja"
  - City: "surat"
  - Email\_ID: "mansi@gmail.com"
  - Name: "Mansi"
  - Phone\_Number: "8641816213"
- M2b7rPeFwX4h4oqjUjl
  - Address: "20, amardeep soociety, katargam, surat"
  - City: "Surat"
  - Email\_ID: "harshi@gmail.com"

Figure 5.5.5 (a) Input data in Firebase

- AWS Lambda Python Function to read data from firebase and write to AWS kinesis data stream

```

File Edit Find View Go Tools Window Save Test ▾
Throttle Qualifiers Actions a
Environment λ lambda_function x +
lambda_function
1 From firebase import firebase
2 import json
3 import boto3
4 from datetime import datetime
5 import calendar
6 import random
7 import time
8
9 def lambda_handler(event, context):
10     firebase1 = firebase.FirebaseApplication('https://tryfirebase2-c9eb0.firebaseio.com/', None)
11     result = firebase1.get('/original', None)
12     my_stream_name = "python-stream"
13     kinesis_client = boto3.client('kinesis', region_name='us-east-1')
14     p_key='partition'
15     payload = {
16         'Data': result
17     }
18     # payload = result
19     print(payload)
20     put_response = kinesis_client.put_record(StreamName=my_stream_name,Data=json.dumps(payload),PartitionKey=p_key)
21     return {
22         'statusCode': 200,
23         'body': 'Successfully Completed'
24     }

```

**Figure 5.5.5 (b) AWS Lambda Function**

- Output file stored in S3 in CSV format by reading the required data

**kinesisresults**

Overview Properties Permissions Management Access points

Type a prefix and press Enter to search. Press ESC to clear.

Upload Create folder Download Actions US East (N. Virginia) Viewing 1 to 1

Name	Last modified	Size	Storage class
abc.csv	Mar 17, 2020 5:51:27 PM GMT+0530	523.0 B	Standard

Viewing 1 to 1

**abc - Excel**

File Home Insert Page Layout Formulas Data Review View Developer Help Tell me what you want to do

A	B	C	D	E	F	G	H	I	J	K
	0	Address	City	Email_ID	Name	Phone_Number				
2	=M2b34OVUg3kzXTnt8CQ	45,dipanje surat		khushali@khushali		8.57E+09				
3	#NAME?	78, manin surat		borad67@nency		8.56E+09				
4	#NAME?	1008 - Lux surat		adev@gmAnuj		9.88E+09				
5	#NAME?	Udhna Dai surat		mansi@grMansi		8.64E+09				
6	#NAME?	20, amard Surat		harshi@grharshita		6.57E+10				
7										
8										
9										
10										

**Figure 5.5.5 (c) Output CSV file in S3**

## 5.6 Test Cases

Test cases for all the activities of projects required are given below in:

**Table 5.6 Test Cases**

<b>Case</b>	<b>Test Data</b>	<b>Expected Result</b>	<b>Actual Result</b>	<b>Pass/Fail</b>
Flutter Login and Registration	- Internet Connectivity - Email ID - Password	If Email ID & password is empty or invalid and No Internet connectivity then display Error message otherwise login successfully	If E-mail ID & password is empty or invalid and no Internet connectivity then displays the error message.	Pass
AWS Customer Activity Tracker	-FlaskForm - CSV file	In both, the task the data need to be read and store successfully at destination	The data were stored in the required format successfully	Pass
AWS Kinesis with Firebase	Firebase Realtime Database	In the task, the data need to be read and write successfully at destination	The data are read from the firebase and written to kinesis stream	Pass
Django Forgot password	Email Id	If email-id is valid & authorized then a new password is set by received email.	Password is reset if the email-id is authorized and valid.	Pass
Flutter Quiz App	Questions of various language	The user should be able to get the results as per the quiz selected without error	The user gets the required results as expected	Pass

**Table 5.6 Test Cases**

<b>Case</b>	<b>Test Data</b>	<b>Expected Result</b>	<b>Actual Result</b>	<b>Pass/Fail</b>

Flutter Registration	- Internet Connectivity - Email ID - Password - User Name	If Email ID & password or username empty or invalid and No Internet connectivity then display Error message otherwise register successfully	If E-mail ID & password and username is empty or invalid and no Internet connectivity then displays the error message.	Pass
AWS Cloudtrail trails	-Create Cloud Trails -JSON file	In the task, the data need to be read and store successfully at destination	The data were stored in the required format successfully	Pass
AWS Athena	-CSV File	In the task, the data need to be read from the CSV file and load in the Athena for analysis	the data to be read from CSV file and load in the Athena for analysis done successfully	Pass
Django Registration Login	-Email Id -Password -Username	If the user name and email id are not valid than OTP is not sent or OTP is invalid no login	Email id are not valid than OTP is not sent or OTP is invalid no login of an unauthorized user	Pass

## CHAPTER 6 CONCLUSION AND FUTURE SCOPE

This internship report stresses on the work experience we have gathered as an Intern at Biz Insights IT Solutions Surat started in December 2019. In this report, we have incorporated our experiences where we worked on various Data Science and Analysis Projects, providing the prototype for Django Hotel Management System also performing the various task with AWS Services such as Lambda, S3, Dynamo DB, API gateway, AWS Kinesis, AWS Athena and other various activities includes Flutter App development, Firebase under our mentor and Co-founder of the Company Mr. Kunal Shah for 6 months.

We can honestly say that our time spent interning with the company resulted in one of the best working experiences of our life. Not only did we gain practical skills but we also had the opportunity to meet many fantastic people. The atmosphere at the office was always welcoming which made us feel right at home. Additionally, we felt like we were able to contribute to the company by working on the projects as well as writing documents as per client requirements on each of the modules performed in the project or training activities.

Whether it was the project proposed to us of the Web Scraping of the 4200+ car dealers of reviews and ratings which was crucial and important client project for the company and believe us that we can work on such new innovative concept and under his guidance we were able to learn the bunch of AWS services and finally implementing the task as mentioned for various project modules. Further, we also worked and deal with various other areas such as web and mobile app development technologies like Django and Flutter Framework and also in the COVID-19 pandemic designing the COVID-19 Patient Prediction System with Machine Learning to help the patients and for future use. Lastly, we all will like to mention that also upcoming the challenges and errors in all the training activities and projects help us to grow in order to get the final results even better. So it was the privilege to gain knowledge and finding the future aspects of many such new areas and technologies in the real-world through this opportunity offered to us.

Overall, our internship at Biz Insights IT Solutions Surat has been a success. We were able to gain practical skills, work in a fantastic environment and, making connections for a lifetime. We are very much thankful to all.

## CHAPTER 7 REFERENCES

### Web references

- [1] [Online]: <https://www.youtube.com/watch?v=XQgXKtPSzUI> for web scraping [Date Accessed: 26 Dec 2019].
- [2] [Online]: <https://data-flair.training/blogs/data-science-project-ideas/> for web scraping and data analysis [Date Accessed: 29 Jan 2020].
- [3] [Online]: <https://data-flair.training/blogs/data-science-project-ideas/> for web scraping and data analysis [Date Accessed: 29 Jan 2020].
- [4] [Online]<https://towardsdatascience.com/amazon-vs-flipkart-finding-the-best-pricesfor-books-bab29811b801> [Date Accessed: 23 Feb 2020].
- [5] [Online] <https://towardsdatascience.com/> [Date Accessed: 25 March 2020].
- [6] [Online]<https://towardsdatascience.com/fight-covid-19-with-machine-learning1d1106192d84> [Date Accessed: 13 March 2020].
- [7] [Online]<https://aws.amazon.com/athena/> [Date Accessed: 29 March 2020].
- [8] [Online]<https://docs.aws.amazon.com/lambda/latest/dg/with-kinesis-example.html> [Date Accessed: 27 April 2020].

## Industrial Training Report