SAAVAAD

A Project / Dissertation as a Course requirement for

BSc Computer Science (Hons)

KODI ROHITH
204204

SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)

Department of Mathematics and Computer Science

Muddenahalli Campus

April, 2023

Dept. of **Mathematics & Computer Science**
Muddenahalli Campus

CERTIFICATE

This is to certify that this Project / Dissertation titled **"SAAVAAD"** Submitted by **KODI ROHITH, 204204,** Department of Mathematics and Computer Science, Muddenahalli Campus is a Bonafide record of the original work done under my supervision as a Course requirement for the Degree of Bachelor of Science (Hons) in Computer Science.

Dr. Sampath Lonka

Project Supervisor

Countersigned by

Head of the Department

Place: Muddenahalli

Date: April 27, 2023

DECLARATION

The Project / Dissertation titled **SAAVAAD** was carried out by me under the supervision of Dr. Sampath Lonka, Department of Mathematics and Computer Science, Muddenahalli Campus as a course requirement for the Degree of Bachelor of Science (Hons) in Computer Science and has not formed the basis for the award of any degree, diploma or any other such title by this or any other University.

K. Rohith

Place: Muddenahalli

KODI ROHITH
204204

Date: 27th April, 2023
Muddenahalli Campus

ACKNOWLEDGEMENT

I would like to extend my sincere thanks to Dr. Sampath Lonka for his supportive, thoughtful and caring guidance, for this project would not be what it is without him.

I would like to extend my gratitude to Bhagawan Sri Sathya Sai baba for giving me the opportunity to study computer science, for I would not have any knowledge without him.

I would also like express my heartfelt thanks to my parents who kept me motivated me throughout the journey of our project in the all the up’s and down’s.

I would also like to thank my Goenka Lokesh and Thilak D for their productive work in the development of this project.

I would also thank each and everyone who played a major and minor role in finishing my project, I would also thank the latest generation AI ChatGPT which also assisted me in various situations of our journey.

TABLE OF CONTENTS

ABSTRACT

In the fast-emerging world, people started with invention, came till implementation which are very helpful in their busy hectic schedule. Following the tradition of our generation, here we came up with a new idea which would help millions of people and especially the student community in majority.

Unveiling our product…" SAAVAAD" is an android along with web application which would assist humanity in language translation and help people in reading big books in short span. This may seem something simple to you, but we would let you how know it works.

Our application would generate summary for a big book or a novel in within few minutes, there is no need for the user to read the entire book and novel to read to understand the gist of a novel or a book, our application could even recognize handwritten text from an image and generate summary for the recognized text.

It would also assist user with audio service which will help user to listen to the summarized text audio again and again like a song, and would help students to remember an answer in a simpler way, how student remember a song.

CHAPTER 1
INTRODUCTION
SAAVAAD

Saavaad is an android application along with website that helps people to summarize and translate their text on their mobile and as well as in their personal computers. The website and app provide an easy and convenient way for the people especially younger generation students to get their summarized text and translated text and save it for future use with title, date and time created.

It is designed to be user-friendly, with a simple and intuitive interface that allows students to access their saved summarized text and translated text quickly and easily.

Lot of things happen in the backend when a user is using our services, the input being entered by the user is sent to the respective API, where it receives the input in the form of request and responds with a response needed by the user.

As the machine learning models are heavier to use, and implement in an android application, we use an API built using flask which is hosted globally by serveo.net organization with our customized URL for our API.

CHAPTER 2
Software Requirement Analysis

2.1 System Features

Features play a crucial role in any application. More and more features need to be provided to the user, and that only would make user to get attracted to any application and use it. Below are some of the features which are necessary for our application.

User Authentication and Authorization: The system will provide user authentication and authorization, ensuring that only authorised users can access the system and perform any action.

Data Retrieval and Storage: The system will retrieve data from various sources, such as databases or external APIs, and store it in the appropriate format for use in the machine learning models.

Prediction/Recommendation: The system will use the trained machine learning models to make predictions or recommendations based on the input data.

Backend API: The system will provide a RESTful API for the website and mobile app to communicate with the backend, handling requests and responses.

Web Application: The system will provide a web-based interface for users to interact with the system, allowing them to upload data, select machine learning models, and view the results.

Mobile Application: The system will provide a mobile-based interface for users to interact with the system on-the-go, allowing them to access the same features as the web application.

2.2 Functional Requirements

Functional requirements describe what and how a software application or product must do to meet the users or stakeholders need. They specify functions, capabilities, and features that the developed product should have to perform its intended tasks.

2.2.1 Website:

The website should have a user interface that allows users to interact with the system.

The website should be able to send requests to the backend server through RESTful APIs.

The website should be able to receive responses from the backend server and display them to the user.

The website should have a login system to authenticate users. e. The website should be able to handle errors gracefully and provide error messages to the user.

2.2.2 Mobile App:

The mobile app should have a user interface that allows users to interact with the system.

The mobile app should be able to send requests to the backend server through RESTful APIs.

The mobile app should be able to receive responses from the backend server and display them to the user.

The mobile app should have a login system to authenticate users.

The mobile app should be able to handle errors gracefully and provide error messages to the user.

2.2.3 Backend Server:

The backend server should be able to receive requests from the website and mobile app through RESTful APIs.

The backend server should be able to process the requests using machine learning models.

The backend server should be able to send responses back to the website and mobile app through RESTful APIs.

The backend server should be scalable and able to handle multiple requests simultaneously.

2.2.4 Machine Learning Models:

The machine learning models should be trained using Python libraries such as Scikit-Learn or TensorFlow.

The machine learning models should be able to process the requests and generate responses.

The machine learning models should be integrated with the backend server using REST APIs.

2.3 Non-Functional Requirements:

Non-Functional requirements are the qualities or attributes that a product must have, but do not relate to the functionality of the system. These requirements are typically concerned with how well the system works, rather than what it does.

Performance:

The system should be able to handle a large number of users and requests simultaneously without any significant delays or slowdowns.

The system should have a fast response time, with requests processed within a few seconds.

Security:

The system should have a secure login system to authenticate users.

The system should use HTTPS to encrypt data sent between the website, mobile app, and backend server.

The system should have measures in place to prevent unauthorised access or attacks on the system.

Scalability:

The system should be designed to handle an increasing number of users and requests as the system grows.

The system should be able to scale horizontally by adding more servers to handle the load.

Usability:

The website and mobile app should have a user-friendly interface that is easy to use and navigate.

The website and mobile app should have clear and concise instructions for the user.

Required Hardware Configuration:

Processor: Intel Dual Core.

Base memory size: 4 GB RAM.

Software Specification:

Operating System: Windows, Linux, macOS, or any other supported by React and React Native.

Development tools: Node.js, React, React Native, Flask and a database management system (e.g., MongoDB).

CHAPTER 3
APLLICATION STUDY

Summarization:

Summarization in layman term means to sumup everything in short but in a detailed manner. In current scenario everyone wishes to learn as fast as possible but at the same they don't want to bear the cost of losing the main content.

There are many ways in which text summarization could be done and many platforms which could do it comfortably.

Mentioning a few which we found:

- SummarizeBot
- SummarizeThis
- TextTeaser
- Resoomer
- SMMRY

But the reason behind we choosing this project is huge amount of work could be done in short period with the help of our beloved machines and their capability to learn and implement what they learn. Though they are not efficient enough as humans, but they could perform the same task n number of times in exact manner as they did earlier in less time. The advantage with this is more amount of output could be generated within less time.

Language translation:

During our study period with this project we found that there might be circumstances where people may feel that not only English, if we could provide summarized text for other languages also.

Thinking on these lines we came up with an idea why can't we try that, there might be many applications which would do English work but, making a pipeline for this may look simple but still people haven't tried it.

Text Recognition & Voice Output:

In the current scenario people aren't even getting the time to type a text or copy paste a text from the source to the application.

So, it's the company's or product developer's responsibility to provide the comfort for the user to make the product user friendly.

We have taken a step ahead to utilise the resources already available to take up this task. Text recognition from images are available and using these we could recognize the text and send it for text summarization to get the summary of a text using the summarization tool.

If the summary is large enough so that user could read it, we also have provided an option to listen to the output.

Proposed idea:

Then we came up with an idea of combining summarization and translation in one application in which user can use our mobile application or web application depending on his choice.

The Summarized rand translated results can be stored locally in his mobile app with created time and date and title of the content and later he can re visit the content when he is in offline.

Pros:

- User friendly – a user can access our application and finish his work whenever possible in a simple and understandable manner.
- Storage – a user can store his data and have a glance at it if he desires to reuse previous collections of data.
- Privacy – a most important aspect which users a very concerned about. A user doesn’t want’s his password to be stored with the company as well, so password encryption and decryption a crucial.

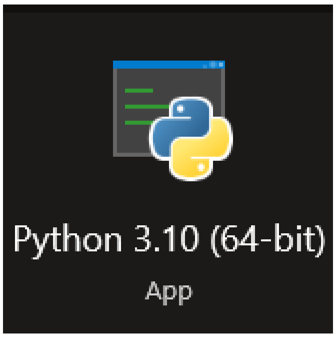
Cons:

- Accuracy – it depends on the models how accurate they perform the task, though the developer may choose the best machine learning models available in the market, it depends entirely on the model to generate an apt summary or translation for a given input.
- Internet – there might be situations where a user may desire to get output without internet, but it’s highly impossible (not impossible but the model to run on the application would make the heavier) because the requested source may not available without internet.

CHAPTER 4
APPLICATION DESIGN

4.1 Language Used

4.1.1 What is Python (in Machine Learning)?



Python is a preferred programming language because of its many capabilities, applicability, and simplicity. Python is the programming language that works well with machine learning.

Python is a programming language that stands out from other programming languages by its flexibility, simplicity, and reliable tools required to construct modern software.

It also stands out from other programming languages by its independent platform and popularity in the programming world. Python is best suited for machine learning since it is reliable and based on simplicity.

4.1.2 Why Python is Most Suitable for Machine Learning

Due to the demand for automation, machine learning and AI as a whole are rapidly expanding in utilisation. Innovative solutions to everyday issues can be developed thanks to artificial intelligence, including fraud detection, personal assistants, spam filters, search engines, and recommendation systems.

The requirement for intelligent answers to practical issues needs the further development of AI in order to automate laborious operations that would be difficult to programme without AI. The Python programming language is thought to be the ideal technique for automating these processes since it is more straightforward and consistent than other programming languages.

Additionally, the vibrant Python community makes it simple for developers to discuss projects and offer suggestions for improving their code.

4.1.3 What makes Python the best programming language for machine learning and AI?

Here we list few features because of which python is choosen as best programming languages for Machine Learning :

- **Simple and consistent :** Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems.

- **Extensive selection of libraries and frameworks :** To reduce development time, programmers turn to a number of Python frameworks and libraries. A software library is pre-written code that developers use to solve common programming tasks. Python, with its rich technology stack, has an extensive set of libraries for artificial intelligence and machine learning.

  Eg : Kera's, TensorFlow, NumPy, Pandas etc.

- **Platform independence :** Platform independence refers to a programming language or framework allowing developers to implement things on one machine and use them on another machine without any (or with only minimal) changes. One key to Python's popularity is that it's a platform independent language. Python is supported by many platforms including Linux, Windows, and macOS.

- **Great community and popularity :** In the Developer Survey 2020 by Stack Overflow, Python was among the top 5 most popular programming languages, which ultimately means that you can find and hire a development company with the necessary skill set to build your AI-based project.

4.2 Machine Learning Concepts

Due to the demand for automation, machine learning and AI as a whole are rapidly expanding in utilisation. Innovative solutions to everyday issues can be developed thanks to artificial intelligence, including fraud detection, personal assistants, spam filters, search engines, and recommendation systems.

The requirement for intelligent answers to practical issues needs the further development of AI in order to automate laborious operations that would be difficult to programme without AI.

The Python programming language is thought to be the ideal technique for automating these processes since it is more straightforward and consistent than other programming languages.

Additionally, the vibrant Python community makes it simple for developers to discuss projects and offer suggestions for improving their code.

4.2.1 Libraries

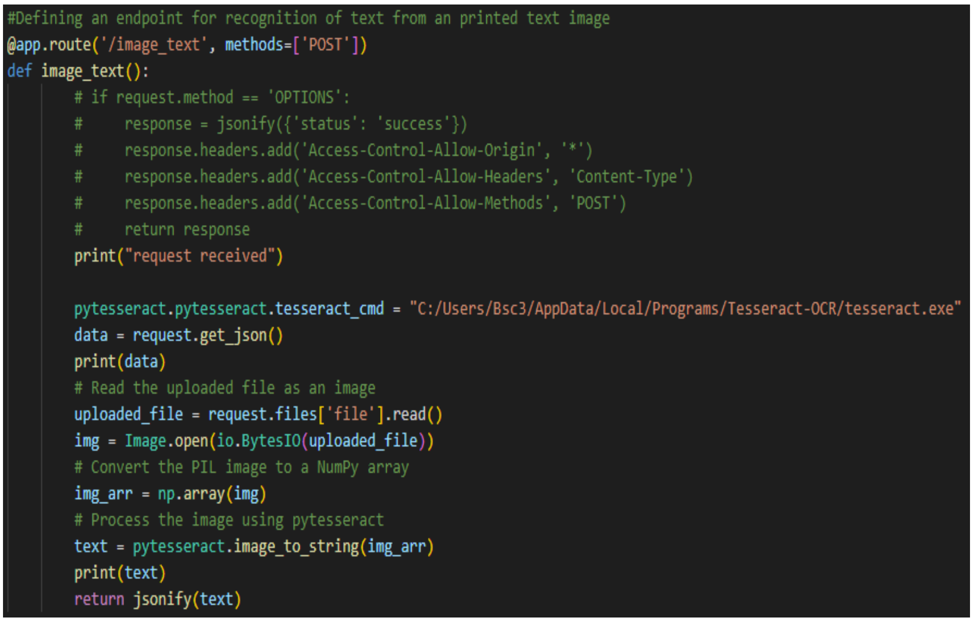
Pytesseract

Pytesseract is a Python library or package that provides an interface for using Tesseract-OCR engine, which is an optical character recognition engine that can be used to recognize text from images.

- Pytesseract provides a simple and easy-to-use Python wrapper around the Tesseract-OCR engine, allowing users to extract text from images in just a few lines of Python code.

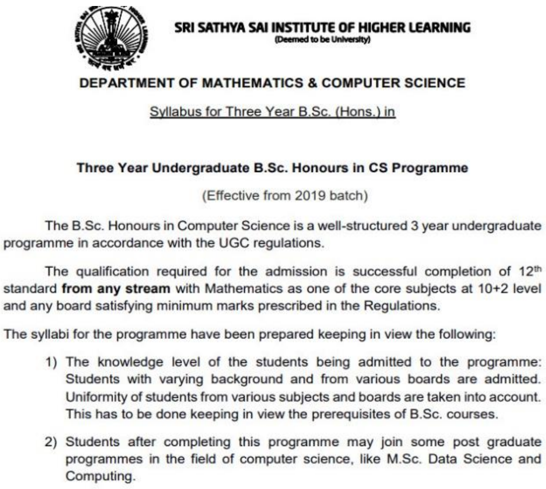
- It supports various image formats such as PNG, JPEG, GIF, TIFF, and BMP, and can be used to extract text from scanned documents, receipts, screenshots, and other types of images.

- Pytesseract is an open-source library and can be installed using pip, the Python package manager.



```python
#Defining an endpoint for recognition of text from an printed text image
@app.route('/image_text', methods=['POST'])
def image_text():
        # if request.method == 'OPTIONS':
        #     response = jsonify({'status': 'success'})
        #     response.headers.add('Access-Control-Allow-Origin', '*')
        #     response.headers.add('Access-Control-Allow-Headers', 'Content-Type')
        #     response.headers.add('Access-Control-Allow-Methods', 'POST')
        #     return response
        print("request received")

        pytesseract.pytesseract.tesseract_cmd = "C:/Users/Bsc3/AppData/Local/Programs/Tesseract-OCR/tesseract.exe"
        data = request.get_json()
        print(data)
        # Read the uploaded file as an image
        uploaded_file = request.files['file'].read()
        img = Image.open(io.BytesIO(uploaded_file))
        # Convert the PIL image to a NumPy array
        img_arr = np.array(img)
        # Process the image using pytesseract
        text = pytesseract.image_to_string(img_arr)
        print(text)
        return jsonify(text)
```

Uploaded image



SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)

DEPARTMENT OF MATHEMATICS & COMPUTER SCIENCE

Syllabus for Three Year B.Sc. (Hons.) in

Three Year Undergraduate B.Sc. Honours in CS Programme

(Effective from 2019 batch)

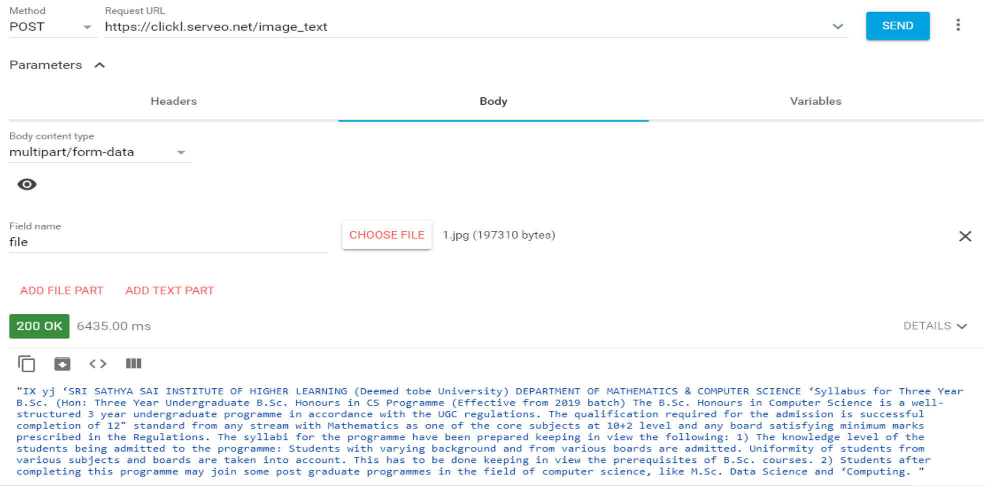
The B.Sc. Honours in Computer Science is a well-structured 3 year undergraduate programme in accordance with the UGC regulations.

The qualification required for the admission is successful completion of 12th standard **from any stream** with Mathematics as one of the core subjects at 10+2 level and any board satisfying minimum marks prescribed in the Regulations.

The syllabi for the programme have been prepared keeping in view the following:

1) The knowledge level of the students being admitted to the programme: Students with varying background and from various boards are admitted. Uniformity of students from various subjects and boards are taken into account. This has to be done keeping in view the prerequisites of B.Sc. courses.
2) Students after completing this programme may join some post graduate programmes in the field of computer science, like M.Sc. Data Science and Computing.

Output from the API



Method: POST

Request URL: https://clickl.serveo.net/image_text

Body content type: multipart/form-data

Field name: file

1.jpg (197310 bytes)

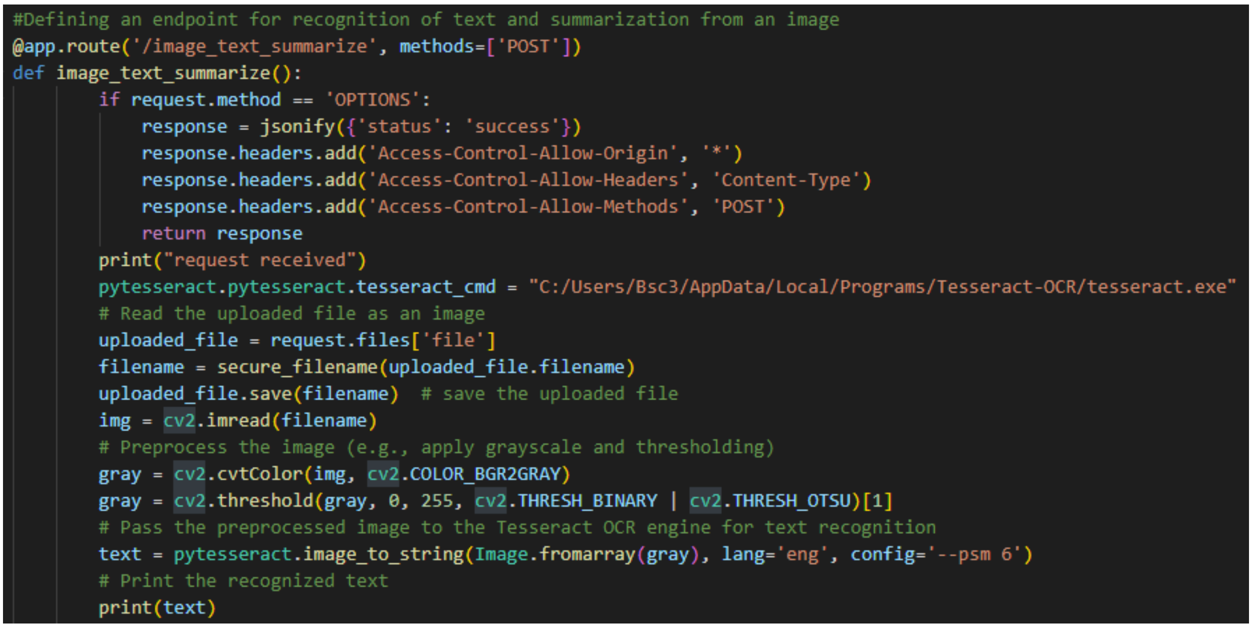
200 OK 6435.00 ms

```
"IX yj ‘SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING (Deemed tobe University) DEPARTMENT OF MATHEMATICS & COMPUTER SCIENCE ‘Syllabus for Three Year
B.Sc. (Hon: Three Year Undergraduate B.Sc. Honours in CS Programme (Effective from 2019 batch) The B.Sc. Honours in Computer Science is a well-
structured 3 year undergraduate programme in accordance with the UGC regulations. The qualification required for the admission is successful
completion of 12" standard from any stream with Mathematics as one of the core subjects at 10+2 level and any board satisfying minimum marks
prescribed in the Regulations. The syllabi for the programme have been prepared keeping in view the following: 1) The knowledge level of the
students being admitted to the programme: Students with varying background and from various boards are admitted. Uniformity of students from
various subjects and boards are taken into account. This has to be done keeping in view the prerequisites of B.Sc. courses. 2) Students after
completing this programme may join some post graduate programmes in the field of computer science, like M.Sc. Data Science and ‘Computing. "
```

Cv2

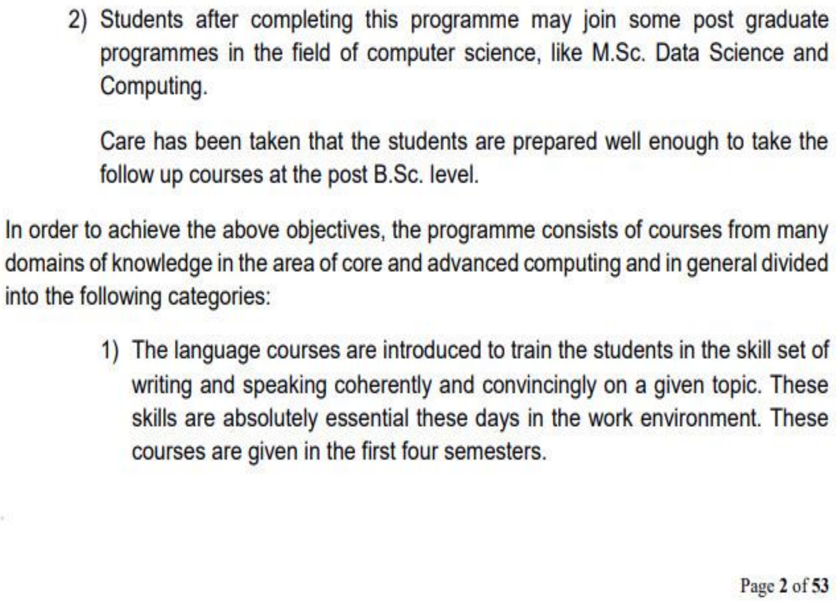
cv2 is a Python library or package that provides an interface to the OpenCV (Open Source Computer Vision) library. It is used for various computer vision and image processing tasks such as object detection, image segmentation, face recognition, and more.

- cv2 provides a wide range of functions and algorithms for image and video analysis, including image manipulation, filtering, feature detection, camera calibration, and geometric transformations. It supports various image formats such as PNG, JPEG, BMP, TIFF, and GIF, and can be used to read and write images and videos from files or capture devices such as webcams.
- cv2 is an open-source library and can be installed using pip, the Python package manager. It is widely used in fields such as robotics, surveillance, medical imaging, and more.



```python
#Defining an endpoint for recognition of text and summarization from an image
@app.route('/image_text_summarize', methods=['POST'])
def image_text_summarize():
        if request.method == 'OPTIONS':
            response = jsonify({'status': 'success'})
            response.headers.add('Access-Control-Allow-Origin', '*')
            response.headers.add('Access-Control-Allow-Headers', 'Content-Type')
            response.headers.add('Access-Control-Allow-Methods', 'POST')
            return response
        print("request received")
        pytesseract.pytesseract.tesseract_cmd = "C:/Users/Bsc3/AppData/Local/Programs/Tesseract-OCR/tesseract.exe"
        # Read the uploaded file as an image
        uploaded_file = request.files['file']
        filename = secure_filename(uploaded_file.filename)
        uploaded_file.save(filename)  # save the uploaded file
        img = cv2.imread(filename)
        # Preprocess the image (e.g., apply grayscale and thresholding)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        gray = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
        # Pass the preprocessed image to the Tesseract OCR engine for text recognition
        text = pytesseract.image_to_string(Image.fromarray(gray), lang='eng', config='--psm 6')
        # Print the recognized text
        print(text)
```

Uploaded image



2) Students after completing this programme may join some post graduate programmes in the field of computer science, like M.Sc. Data Science and Computing.

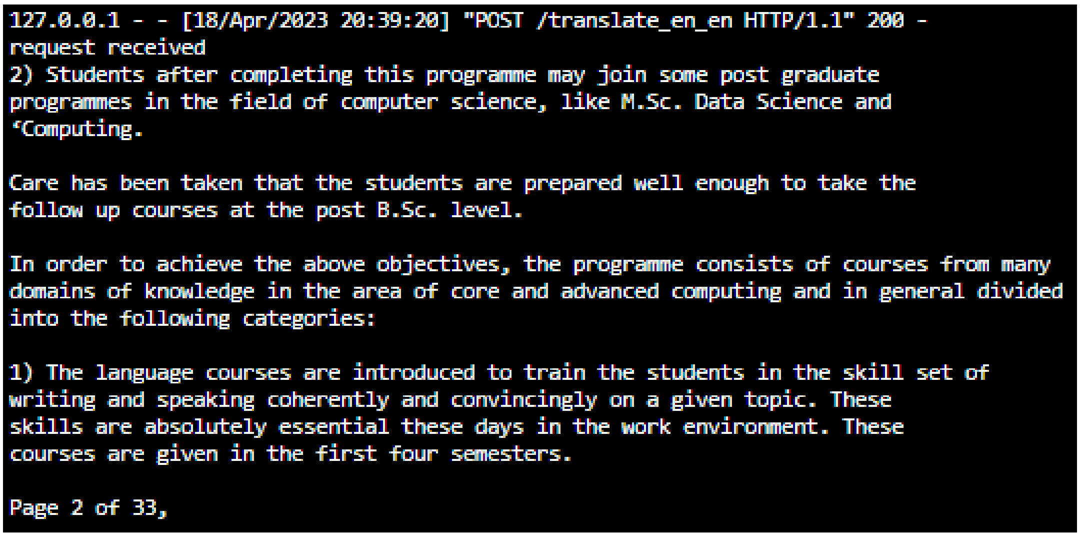
Care has been taken that the students are prepared well enough to take the follow up courses at the post B.Sc. level.

In order to achieve the above objectives, the programme consists of courses from many domains of knowledge in the area of core and advanced computing and in general divided into the following categories:

1) The language courses are introduced to train the students in the skill set of writing and speaking coherently and convincingly on a given topic. These skills are absolutely essential these days in the work environment. These courses are given in the first four semesters.

Page 2 of 53

Recognized text



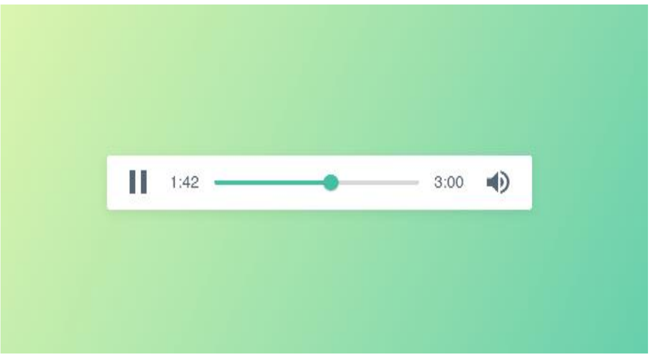
```
127.0.0.1 - - [18/Apr/2023 20:39:20] "POST /translate_en_en HTTP/1.1" 200 -
request received
2) Students after completing this programme may join some post graduate
programmes in the field of computer science, like M.Sc. Data Science and
‘Computing.

Care has been taken that the students are prepared well enough to take the
follow up courses at the post B.Sc. level.

In order to achieve the above objectives, the programme consists of courses from many
domains of knowledge in the area of core and advanced computing and in general divided
into the following categories:

1) The language courses are introduced to train the students in the skill set of
writing and speaking coherently and convincingly on a given topic. These
skills are absolutely essential these days in the work environment. These
courses are given in the first four semesters.

Page 2 of 33,
```

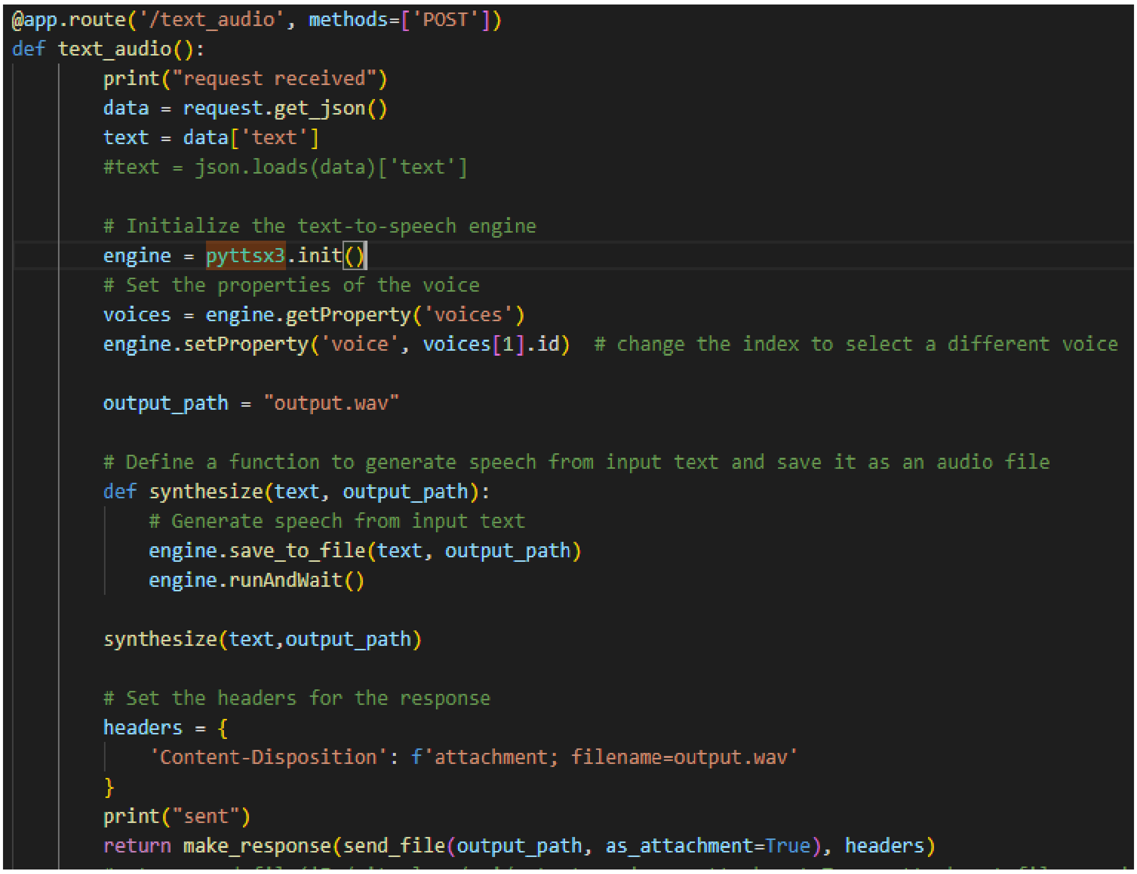
Pyttsx3

pyttsx3 is a Python library or package that is used for text-to-speech conversion. It provides a cross-platform implementation of the Text-to-Speech (TTS) API for Windows, Linux, and macOS.

- pyttsx3 supports various TTS engines such as sapi5, nsss, espeak, and mbrola, and allows users to customize the voice, pitch, rate, and volume of the generated speech. It also provides features such as callbacks, event-driven programming, and thread safety for more advanced usage.
- pyttsx3 can be installed using pip, the Python package manager, and can be used for various applications such as speech synthesis for visually impaired users, language learning tools, voice-enabled chatbots, and more.
- Text being sent :- “Hello, how are you, hope everything is going good. How @bout me reading this text? Is this sounding good? Hurray! I am able to convert text to speech.

Output :



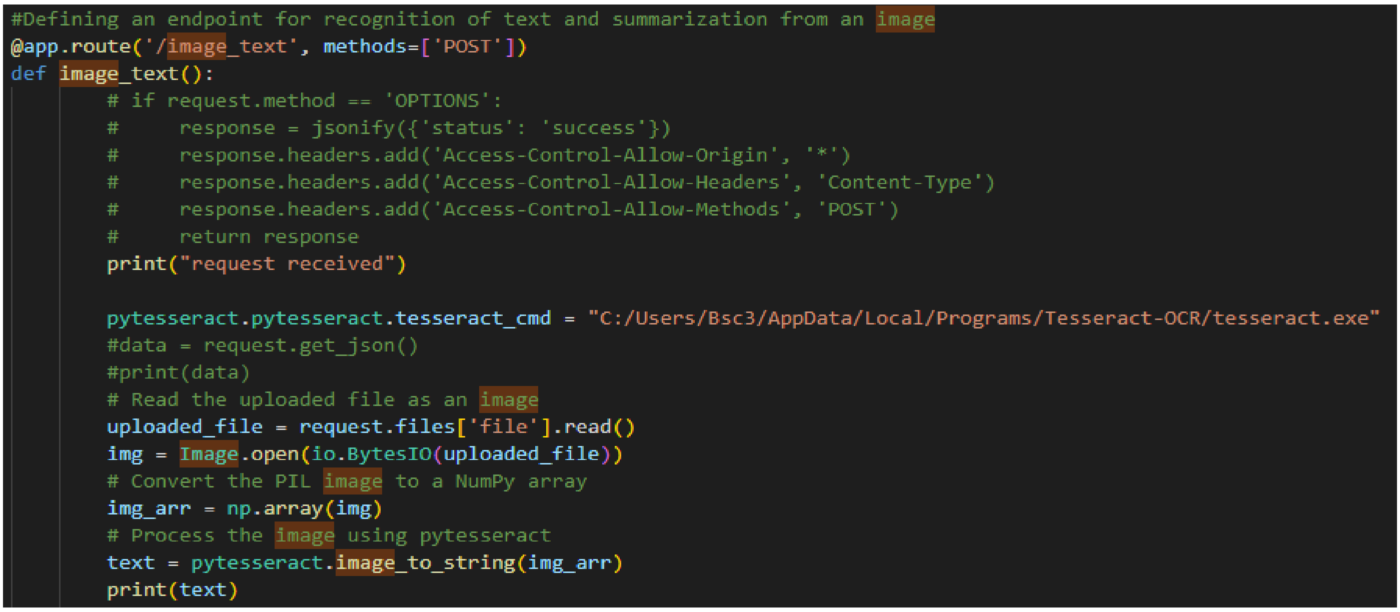


```
@app.route('/text_audio', methods=['POST'])
def text_audio():
        print("request received")
        data = request.get_json()
        text = data['text']
        #text = json.loads(data)['text']

        # Initialize the text-to-speech engine
        engine = pyttsx3.init()
        # Set the properties of the voice
        voices = engine.getProperty('voices')
        engine.setProperty('voice', voices[1].id)  # change the index to select a different voice

        output_path = "output.wav"

        # Define a function to generate speech from input text and save it as an audio file
        def synthesize(text, output_path):
            # Generate speech from input text
            engine.save_to_file(text, output_path)
            engine.runAndWait()

        synthesize(text,output_path)

        # Set the headers for the response
        headers = {
            'Content-Disposition': f'attachment; filename=output.wav'
        }
        print("sent")
        return make_response(send_file(output_path, as_attachment=True), headers)
```

PIL – Python Imaging Library

PIL stands for Python Imaging Library, which is a python library used for opening, manipulating, and saving many different image file formats. It supports a wide range of image file formats including BMP, GIF, JPEG, PNG, TIFF, and others.

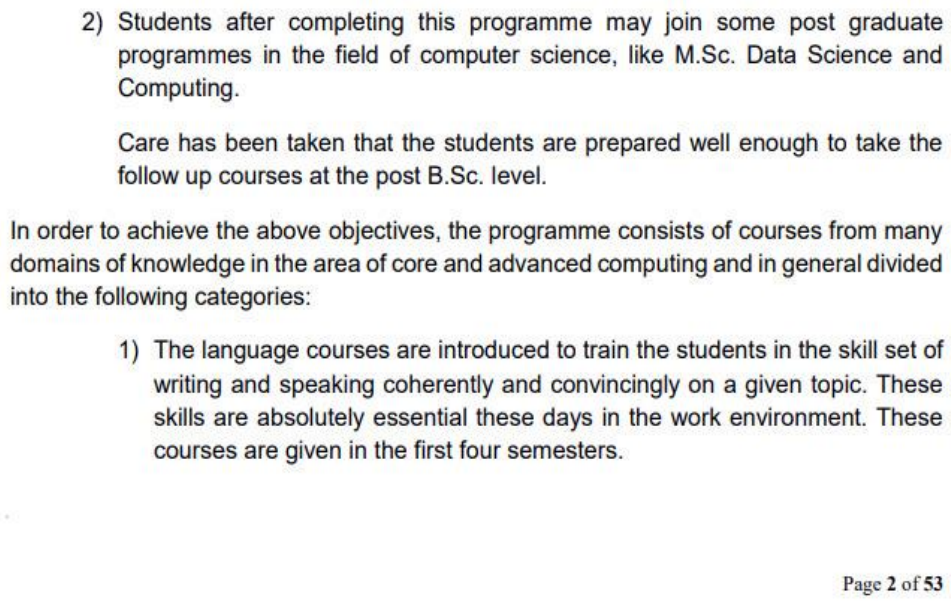
- PIL provides many features for image processing, such as cropping, resizing, color manipulation, adding text, and many others. It is widely used in various fields like image processing, computer vision, machine learning, and data analysis.
- In addition to the basic image processing functionalities, PIL also has additional libraries like ImageDraw, ImageFont, and ImageFilter for drawing and filtering images.



```
#Defining an endpoint for recognition of text and summarization from an image
@app.route('/image_text', methods=['POST'])
def image_text():
        # if request.method == 'OPTIONS':
        #     response = jsonify({'status': 'success'})
        #     response.headers.add('Access-Control-Allow-Origin', '*')
        #     response.headers.add('Access-Control-Allow-Headers', 'Content-Type')
        #     response.headers.add('Access-Control-Allow-Methods', 'POST')
        #     return response
        print("request received")

        pytesseract.pytesseract.tesseract_cmd = "C:/Users/Bsc3/AppData/Local/Programs/Tesseract-OCR/tesseract.exe"
        #data = request.get_json()
        #print(data)
        # Read the uploaded file as an image
        uploaded_file = request.files['file'].read()
        img = Image.open(io.BytesIO(uploaded_file))
        # Convert the PIL image to a NumPy array
        img_arr = np.array(img)
        # Process the image using pytesseract
        text = pytesseract.image_to_string(img_arr)
        print(text)
```

Here an image is opened using PIL and converted to numpy array using numpy(np) and printed text is being extracted from the image and printed.

Uploaded image :



2) Students after completing this programme may join some post graduate programmes in the field of computer science, like M.Sc. Data Science and Computing.

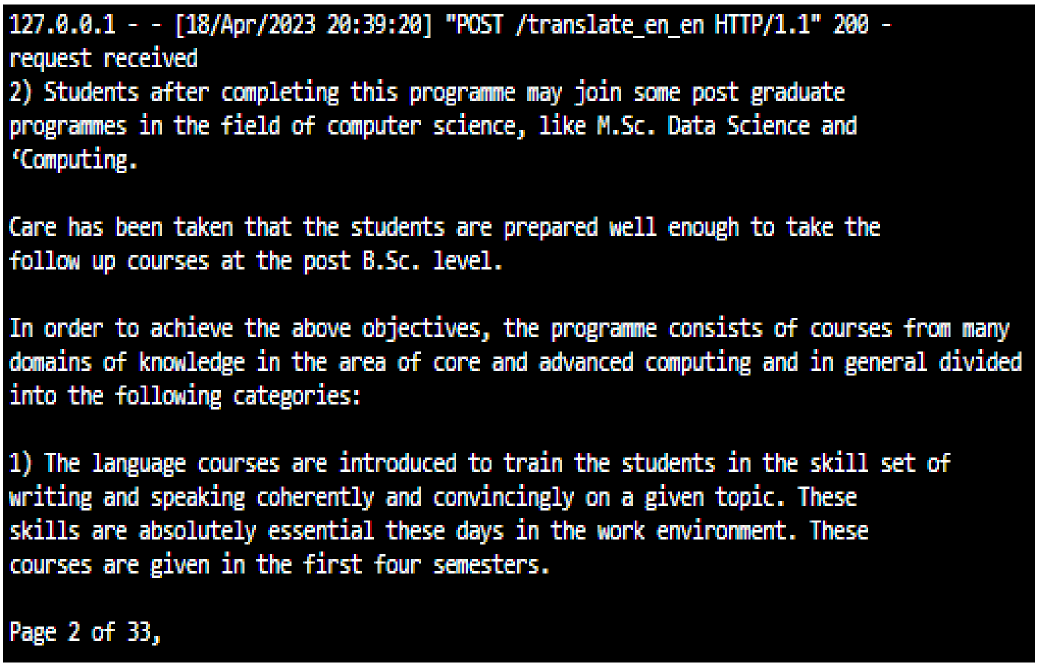
Care has been taken that the students are prepared well enough to take the follow up courses at the post B.Sc. level.

In order to achieve the above objectives, the programme consists of courses from many domains of knowledge in the area of core and advanced computing and in general divided into the following categories:

1) The language courses are introduced to train the students in the skill set of writing and speaking coherently and convincingly on a given topic. These skills are absolutely essential these days in the work environment. These courses are given in the first four semesters.

Page **2** of **53**

Image output :

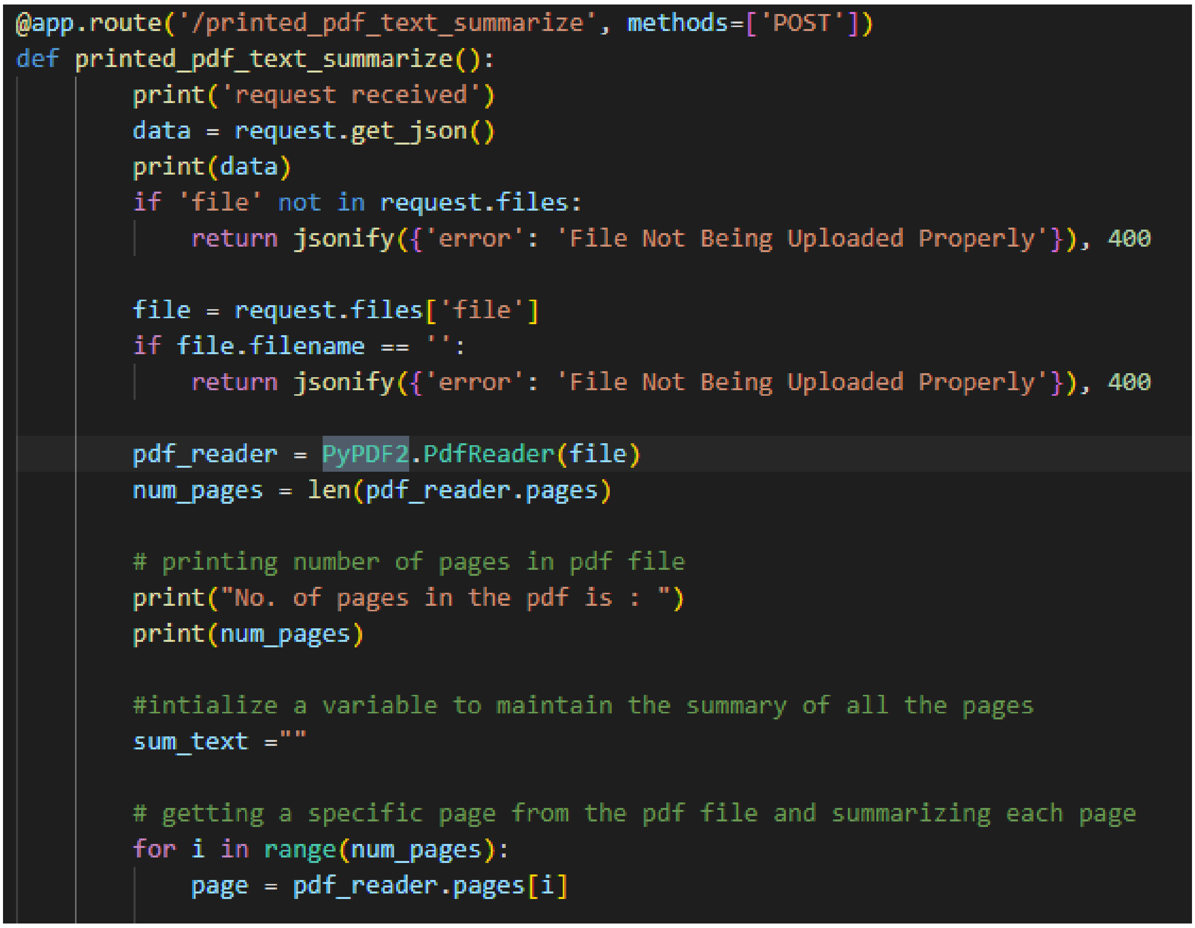


```
127.0.0.1 - - [18/Apr/2023 20:39:20] "POST /translate_en_en HTTP/1.1" 200 -
request received
2) Students after completing this programme may join some post graduate
programmes in the field of computer science, like M.Sc. Data Science and
‘Computing.

Care has been taken that the students are prepared well enough to take the
follow up courses at the post B.Sc. level.

In order to achieve the above objectives, the programme consists of courses from many
domains of knowledge in the area of core and advanced computing and in general divided
into the following categories:

1) The language courses are introduced to train the students in the skill set of
writing and speaking coherently and convincingly on a given topic. These
skills are absolutely essential these days in the work environment. These
courses are given in the first four semesters.

Page 2 of 33,
```

PyPDF2

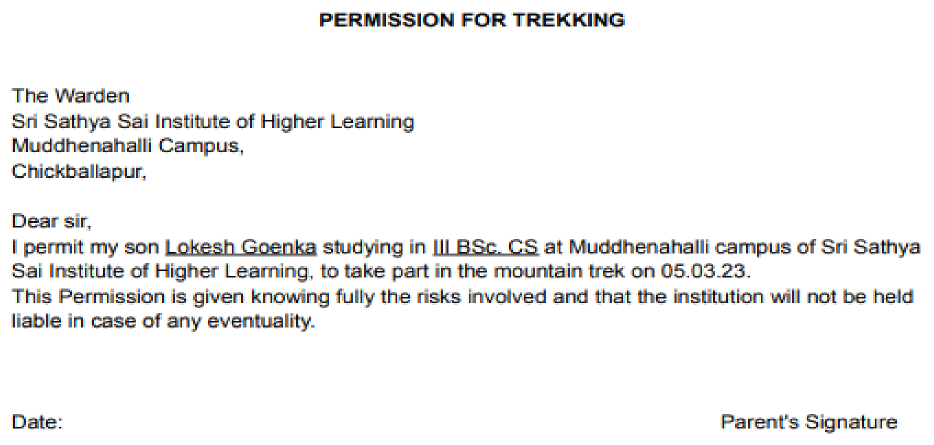
It is a python library used for performing major tasks on PDF files such as extracting the document-specific information, merging the PDF files, splitting the pages of a PDF file, adding watermarks to a file, encrypting and decrypting the PDF files, etc. We will use the PyPDF2 library in this tutorial.

- It is a pure python library so it can run on any platform without any platform-related dependencies on any external libraries.
- It can read, parse and write PDFs. It can be used as a command line tool or as a library.
- It supports Unicode strings so that it can handle non-English characters.



```python
@app.route('/printed_pdf_text_summarize', methods=['POST'])
def printed_pdf_text_summarize():
        print('request received')
        data = request.get_json()
        print(data)
        if 'file' not in request.files:
            return jsonify({'error': 'File Not Being Uploaded Properly'}), 400

        file = request.files['file']
        if file.filename == '':
            return jsonify({'error': 'File Not Being Uploaded Properly'}), 400

        pdf_reader = PyPDF2.PdfReader(file)
        num_pages = len(pdf_reader.pages)

        # printing number of pages in pdf file
        print("No. of pages in the pdf is : ")
        print(num_pages)

        #intialize a variable to maintain the summary of all the pages
        sum_text =""

        # getting a specific page from the pdf file and summarizing each page
        for i in range(num_pages):
            page = pdf_reader.pages[i]
```

Uploaded pdf :



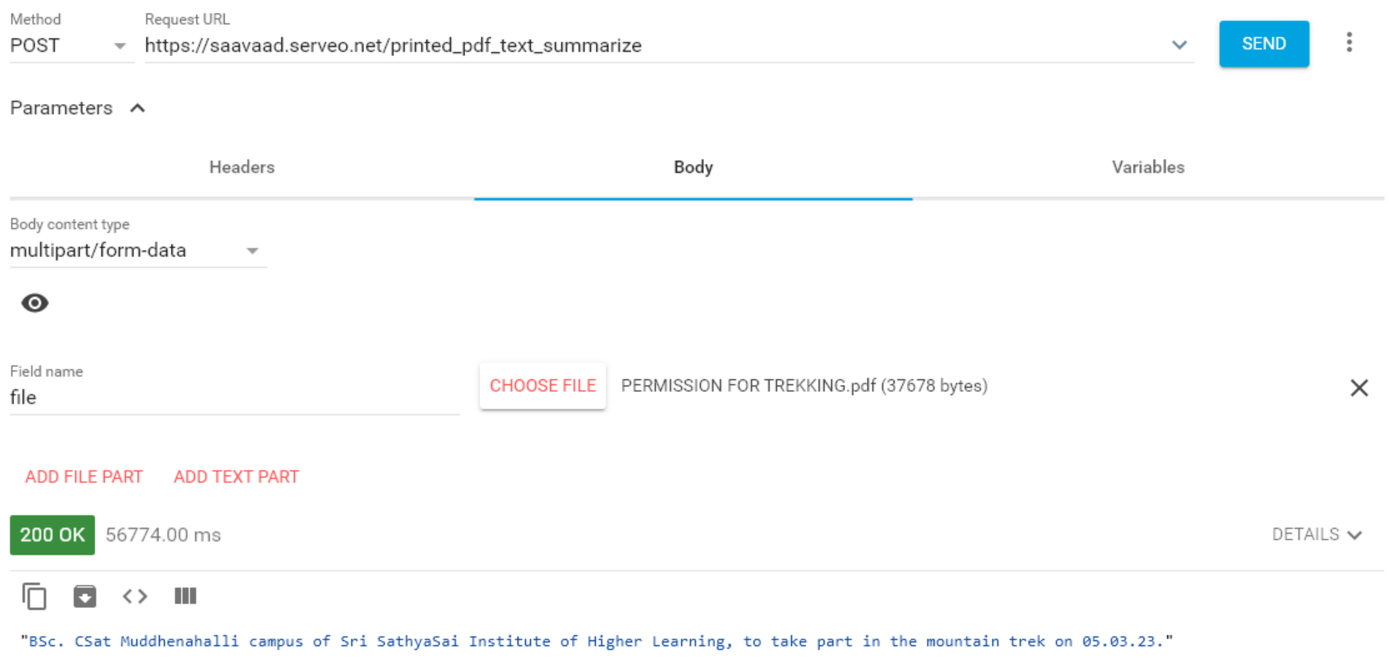
PERMISSION FOR TREKKING

The Warden
Sri Sathya Sai Institute of Higher Learning
Muddhenahalli Campus,
Chickballapur,

Dear sir,
I permit my son Lokesh Goenka studying in III BSc. CS at Muddhenahalli campus of Sri Sathya Sai Institute of Higher Learning, to take part in the mountain trek on 05.03.23.
This Permission is given knowing fully the risks involved and that the institution will not be held liable in case of any eventuality.

Date: Parent's Signature

Output from the API :



Method
POST

Request URL
https://saavaad.serveo.net/printed_pdf_text_summarize

Body content type
multipart/form-data

Field name
file

PERMISSION FOR TREKKING.pdf (37678 bytes)

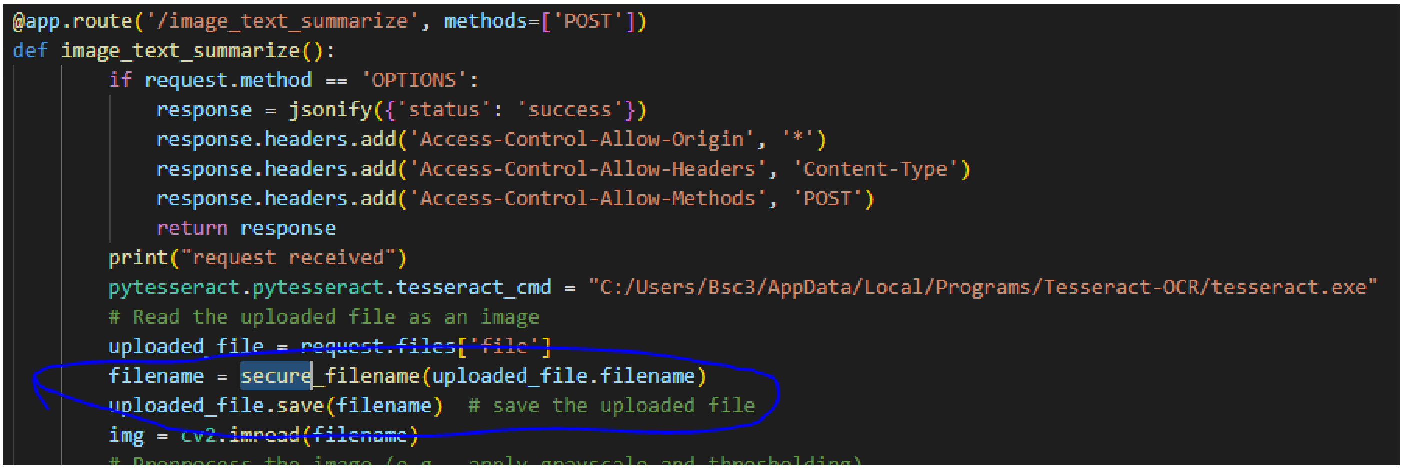
200 OK 56774.00 ms

```
"BSc. CSat Muddhenahalli campus of Sri SathyaSai Institute of Higher Learning, to take part in the mountain trek on 05.03.23."
```

Werkzeug.utils.secure_filename

werkzeug.utils.secure_filename is a function provided by the Werkzeug library, which is a WSGI utility library for Python. It is used to make sure that the filename of a file uploaded to a web server is safe to use, and does not contain any malicious characters or sequences that could be used to execute code or otherwise compromise the server.

- The function replaces any potentially dangerous characters with safe alternatives, and removes any leading dots or slashes from the filename. This helps to prevent security vulnerabilities such as directory traversal attacks, which could allow an attacker to access files on the server outside of the intended upload directory.



```
@app.route('/image_text_summarize', methods=['POST'])
def image_text_summarize():
        if request.method == 'OPTIONS':
            response = jsonify({'status': 'success'})
            response.headers.add('Access-Control-Allow-Origin', '*')
            response.headers.add('Access-Control-Allow-Headers', 'Content-Type')
            response.headers.add('Access-Control-Allow-Methods', 'POST')
            return response
        print("request received")
        pytesseract.pytesseract.tesseract_cmd = "C:/Users/Bsc3/AppData/Local/Programs/Tesseract-OCR/tesseract.exe"
        # Read the uploaded file as an image
        uploaded_file = request.files['file']
        filename = secure_filename(uploaded_file.filename)
        uploaded_file.save(filename)  # save the uploaded file
        img = cv2.imread(filename)
        # Preprocess the image (e.g., apply grayscale and thresholding)
```

Kraken

Kraken is an OCR (Optical Character Recognition) engine that is used to recognize text from scanned images or photographs. It is a free and open-source software that uses deep learning techniques to perform OCR on images. Kraken is highly accurate and can recognize text in multiple languages.

Kraken is designed to be highly modular and customizable, allowing users to fine-tune its OCR capabilities for specific use cases. It also includes a command-line interface and an API, making it easy to integrate into other applications.

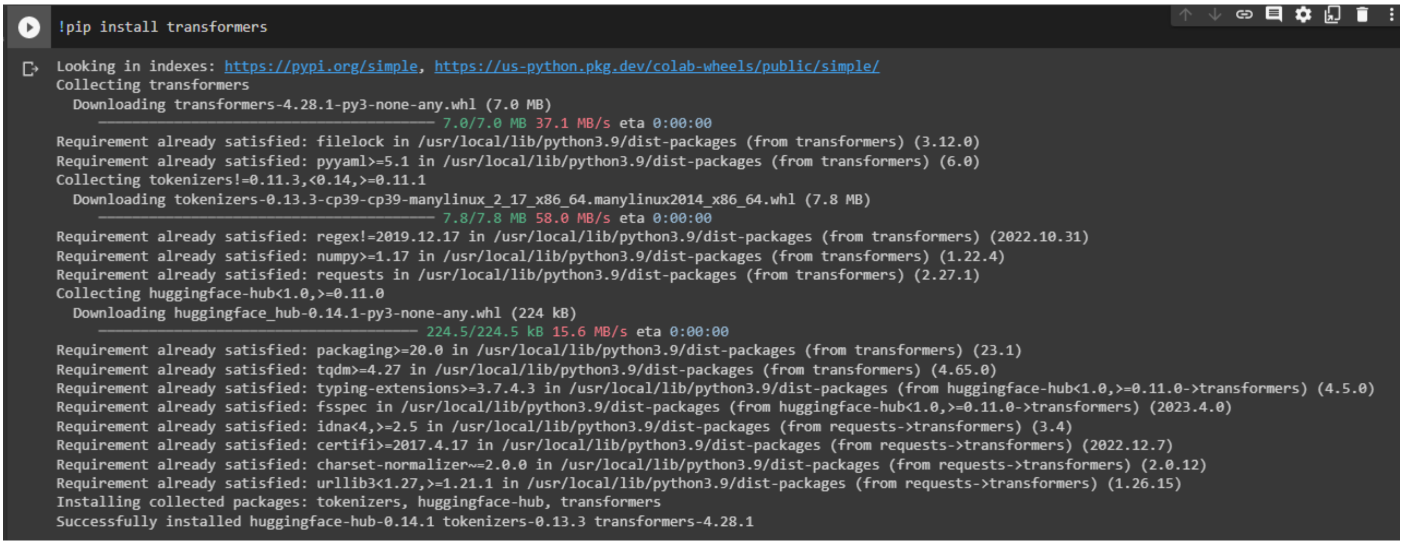
Kraken is commonly used in libraries, archives, and other cultural heritage institutions to digitize their collections and make them more accessible to researchers and the public. It can also be used in industries such as finance, healthcare, and legal services to automate data entry tasks and improve efficiency.

4.2.2 Transformers

Transformers are a type of neural network architecture that is commonly used for natural language processing (NLP) tasks such as language translation, text summarization, question answering, and sentiment analysis. They are named "transformers" because they rely heavily on a self-attention mechanism that can transform the representation of words or tokens within a sentence, allowing the model to focus on different parts of the input text at different times.

Transformers can be trained on large amounts of text data, and then used to generate predictions on new input text. They have achieved state-of-the-art performance on a wide range of NLP tasks and are widely used in industry and academia.

Hugging Face is a popular open-source library for working with pre-trained transformer models in NLP. These models have already been trained on large datasets and can be used as a starting point for fine-tuning on specific tasks or for generating predictions on new text data. To use these models as local models, you can download the pre-trained weights and load them into the appropriate Python framework (such as TensorFlow or PyTorch). Once loaded, you can use the model to make predictions on new input text.



```
!pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting transformers
  Downloading transformers-4.28.1-py3-none-any.whl (7.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.0/7.0 MB 37.1 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.9/dist-packages (from transformers) (3.12.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.9/dist-packages (from transformers) (6.0)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
  Downloading tokenizers-0.13.3-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 58.0 MB/s eta 0:00:00
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.9/dist-packages (from transformers) (2022.10.31)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.9/dist-packages (from transformers) (1.22.4)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from transformers) (2.27.1)
Collecting huggingface-hub<1.0,>=0.11.0
  Downloading huggingface_hub-0.14.1-py3-none-any.whl (224 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 224.5/224.5 kB 15.6 MB/s eta 0:00:00
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from transformers) (23.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.9/dist-packages (from transformers) (4.65.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.9/dist-packages (from huggingface-hub<1.0,>=0.11.0->transformers) (4.5.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.9/dist-packages (from huggingface-hub<1.0,>=0.11.0->transformers) (2023.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->transformers) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->transformers) (2022.12.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->transformers) (2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests->transformers) (1.26.15)
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.14.1 tokenizers-0.13.3 transformers-4.28.1
```

Libraries imported from transformers to use the machine learning models

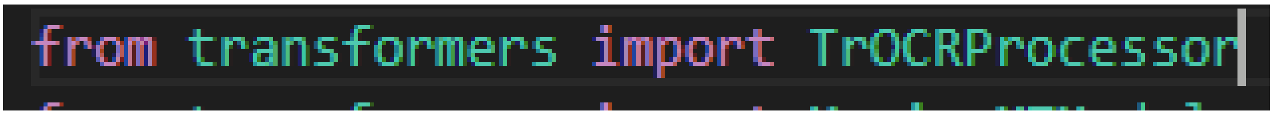
- **TrOCRProcessor**
- **VisionEncoderDecoderModel**
- **MarianMTModel**
- **MarianTokenizer**
- **T5Tokenizer**
- **T5ForConditionalGeneration**
- **T5Config**

TrOCRProcessor :

TrOCRProcessor is a class from the Hugging Face Transformers library that provides text recognition functionality using Optical Character Recognition (OCR) techniques. It pre-processes the input image to prepare it for recognition by splitting it into smaller parts and normalizing them.

Then it feeds the pre-processed images to a pre-trained vision encoder-decoder model that generates a sequence of characters that represents the text in the image. Finally, the TrOCRProcessor class post-processes the output sequence to generate the final recognized text.

The size of the TrOCRProcessor depends on the pre-trained vision encoder-decoder model that it uses. The size can vary from a few MBs to several GBs depending on the complexity and the number of parameters of the model.



```
from transformers import TrOCRProcessor
```

VisionEncodeDecoderModel :

The VisionEncoderDecoderModel is a class from the Hugging Face transformers library that provides a general-purpose architecture for vision-to-sequence tasks.

It combines a vision encoder, such as ResNet, with a sequence decoder, such as a transformer. The model can be fine-tuned for a variety of tasks such as image captioning, visual question answering, and image-to-text generation.

The VisionEncoderDecoderModel is a large model, with a size that depends on the specific encoder and decoder used. For example, the VLP-Base-Image-Text model, which uses a ResNet-101 encoder and a transformer decoder, has over 285 million parameters and a size of around 1.2 GB when stored on disk.

```
from transformers import VisionEncoderDecoderModel
```

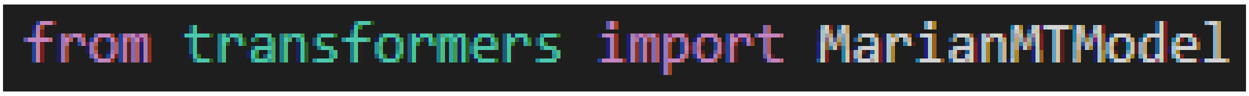
MarianMTModel :

MarianMTModel is a class from the Transformers library which implements the Marian architecture for machine translation.

It is a sequence-to-sequence model that translates text from one language to another. The model is based on the encoder-decoder architecture, where the encoder processes the input sequence and the decoder generates the output sequence.

MarianMTModel is trained on large parallel corpora for various languages. The model is available in different sizes, ranging from small models with a few million parameters to large models with billions of parameters. The size of the model depends on the number of layers, hidden size, and other hyperparameters.

Overall, MarianMTModel is a powerful tool for machine translation, and it can be used for a wide range of applications, such as chatbots, language translation, and content localization.



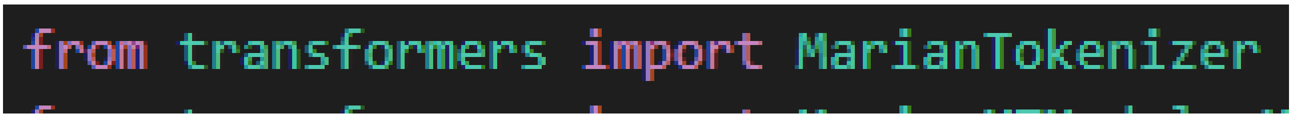
```
from transformers import MarianMTModel
```

MarianTokenizer :

MarianTokenizer is a class in the Transformers library that is used for tokenizing text data. Tokenization is the process of breaking up a piece of text into individual tokens or words, which can be used for various natural language processing tasks such as text classification, named entity recognition, and machine translation.

MarianTokenizer is specifically designed for use with the Marian machine translation models in the Transformers library, and it uses a Byte-Pair Encoding (BPE) algorithm to tokenize text.

The BPE algorithm is a popular technique in natural language processing that works by iteratively merging the most frequently occurring pairs of characters in a text corpus.



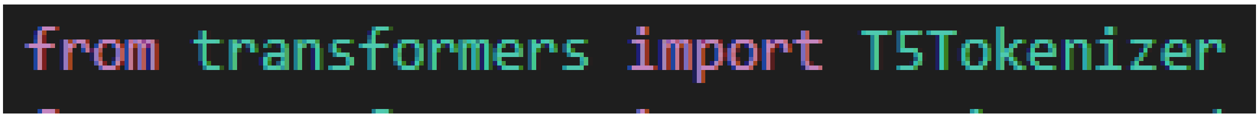
```
from transformers import MarianTokenizer
```

T5Tokenizer :

T5Tokenizer is a tokenizer class provided by the Transformers library, which is used for tokenizing text data for input to the T5 model. It is specifically designed for tokenizing text for summarization tasks.

Tokenization is the process of breaking up the input text into smaller units, known as tokens. These tokens are then converted into numerical vectors, which can be input to a machine learning model like T5.

The T5Tokenizer uses a byte-level Byte-Pair Encoding (BPE) algorithm to convert input text into tokens. BPE is a data compression technique that replaces common sequences of characters with single tokens. This helps to reduce the size of the vocabulary and improve the model's performance.



```
from transformers import T5Tokenizer
```

T5ForConditionalGeneration :

T5ForConditionalGeneration is a pre-trained transformer-based language model from the T5 family of models in the Hugging Face Transformers library. It is specifically designed for text generation tasks, such as text summarization, translation, question answering, and more.

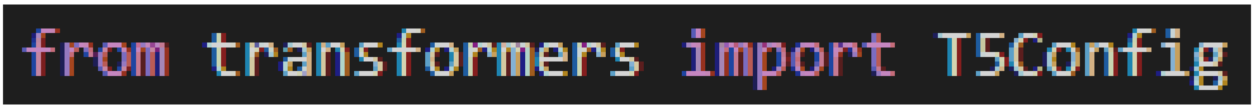
The T5ForConditionalGeneration model works by encoding the input text into a fixed-length vector representation using a series of self-attention layers, and then decoding this representation to generate the output text using another series of self-attention layers.

```
from transformers import T5ForConditionalGeneration
```

T5Config :

T5Config is a class in the Hugging Face Transformers library that is used to store the configuration of the T5 model.

When initializing a T5 model, a T5Config object is passed to define the model's configuration. The T5Config object is used to initialize the weights of the T5 model and determines its behaviour during training and inference.



```
from transformers import T5Config
```

4.2.3 Major Machine Learning Models used :

Machine learning models are algorithms that can automatically learn and make predictions or decisions based on data inputs. These models are built using statistical and mathematical techniques.

They are trained using large amounts of data to identify patterns and relationships. Once trained, these models can make predictions or decisions on new data inputs.

Machine learning models can be used for a wide range of tasks, such as image recognition, speech recognition, natural language processing, fraud detection, recommendation systems, and many others.

They are used in various industries, including healthcare, finance, marketing, and transportation, among others.

- **T5-base (Summarize)**
- **Microsoft/trocr-base-handwritten & Kraken (Recognize)**
- **Helsinki-NLP/opus-mt-(source)-(destination) (Translate)**

T5-base :

T5 is a pre-trained sequence-to-sequence (Seq2Seq) model that was introduced by Google AI Language in 2019. It stands for “Text-to-Text Transfer Transformer” and was built based on the Transformer architecture, which is a deep learning model that uses attention mechanisms to process sequential data. T5 is particularly popular for its versatility, as it can be fine-tuned for a wide range of natural language processing (NLP) tasks such as classification, summarization, translation, and more.

T5 is trained on a large amount of text data and is capable of understanding and generating text in natural language. To perform a specific NLP task using T5, the model is fine-tuned on a smaller dataset that is specific to that task. Fine-tuning involves adjusting the pre-trained weights of the model to make it more accurate and relevant for the specific task.

Overall, T5-base is a powerful tool for natural language processing tasks and is widely used in industry and academia for a variety of applications.

microsoft/trocr-base-handwritten :

‘microsoft/trocr-base-handwritten’ is an OCR (Optical Character Recognition) model released by Microsoft, specifically designed to recognize handwritten text in English. It is a transformer-based model built on the Hugging Face Transformers library and fine-tuned on a large-scale handwriting dataset. The model is capable of recognizing text in various formats, such as scanned images or photos of handwritten notes or documents.

The model works by taking an input image, preprocessing it by resizing and normalizing the image, and then encoding the image into a fixed-length feature vector using a convolutional neural network (CNN). The feature vector is then passed through several layers of transformer blocks, which allow the model to capture contextual information and recognize complex patterns in the text.

Finally, the model decodes the output sequence of characters using a beam search algorithm to generate the most likely output.

Kraken :

Kraken is an optical character recognition (OCR) engine that is used to recognize text from scanned images. It is a machine learning model developed by the University of Erfurt, Germany.

Kraken uses deep learning techniques to perform OCR on scanned images. The model is trained on large datasets of images and corresponding text transcriptions. During training, the model learns to recognize the patterns in the images and associate them with the corresponding text.

Kraken is designed to work with historical and degraded texts, which can be difficult to recognize with traditional OCR engines. It is also highly customizable and can be adapted to recognize specific fonts or scripts.

Helsinki-NLP/opus-mt-(source)-(destination) :

The Helsinki-NLP/opus-mt-(source)-(destination) model is a machine translation model provided by the Helsinki NLP group. It is part of the OpusMT project, which is a collection of machine translation models for a wide range of language pairs.

The model is named after the source and destination languages that it translates between. For example, Helsinki-NLP/opus-mt-en-es is a model that translates from English to Spanish.

To use this model, you would typically input a sentence or a piece of text in the source language, and the model would output the translated text in the destination language.

The model can be used through the Hugging Face Transformers library, which provides a simple Python interface for using pre-trained machine learning models.

4.3 Flask – API

4.3.1 What is an application programming interface (API)?

An API (Application Programming Interface) is a set of protocols, routines, and tools for building software applications. It defines the way in which different software components should interact with each other. APIs allow different software applications to communicate with each other and share data.

An API is a specification for how different software components should interact with each other, while a server is a program or device that provides a service to other programs or devices. An API can be used to access services provided by a server.

4.3.2 How do APIs work?

An API is a set of defined rules that explain how computers or applications communicate with one another. APIs sit between an application and the web server, acting as an intermediary layer that processes data transfer between systems.

Here's how an API works:

- A client application initiates an API call to retrieve information—also known as a request. This request is processed from an application to the web server via the API's Uniform Resource Identifier (URI) and includes a request verb, headers, and sometimes, a request body.
- After receiving a valid request, the API makes a call to the external program or web server.
- The server sends a response to the API with the requested information.
- The API transfers the data to the initial requesting application.

4.3.3 What is an API endpoint and why is it important?

API endpoints are the final touchpoints in the API communication system. These include server URLs, services, and other specific digital locations from where information is sent and received between systems. API endpoints are critical to enterprises for two main reasons:

Security : API endpoints make the system vulnerable to attack. API monitoring is crucial for preventing misuse.

Performance : API endpoints, especially high traffic ones, can cause bottlenecks and affect system performance.

How can everyone access this huge sized ML models :

The only way to access this large data is through API, making a request to API inside which lies all the ml models in its system, and generate endpoints to enter and retrieve data from the API. After entering the API, the input is received by the system and processing id performed on it and the output is returned to the user through the API endpoint.

In our case the systems running the models will become the endpoint server and the user making request to those resources becomes the client.

4.3.4 Flask :

Flask is a lightweight web framework that is often used to build APIs due to its simplicity, flexibility, and ease of use. Here are some advantages and disadvantages of using Flask for building APIs.

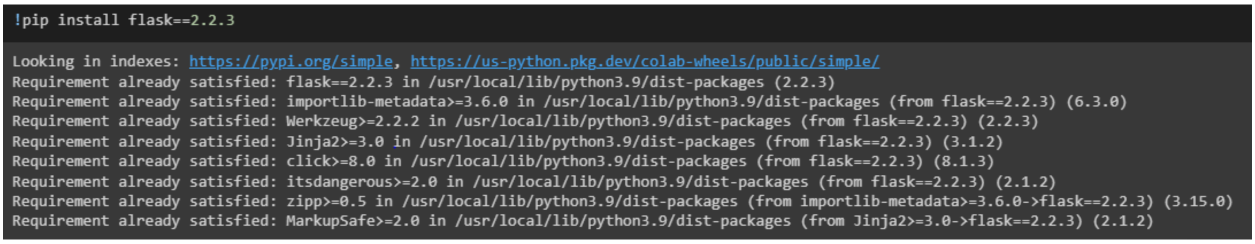
Advantages:

- Flask is easy to set up and configure, making it a great choice for building simple APIs quickly.
- Flask is highly customizable, and allows developers to choose which components to include in their application, resulting in a more lightweight and efficient API.
- Flask has a large and active community, which provides a wealth of resources and plugins to extend its functionality.
- Flask supports a wide range of data formats, including JSON, XML, and CSV, making it a versatile choice for building APIs that work with different types of data.
- Flask provides built-in support for unit testing, making it easy to ensure that your API is working as expected.

Disadvantages:

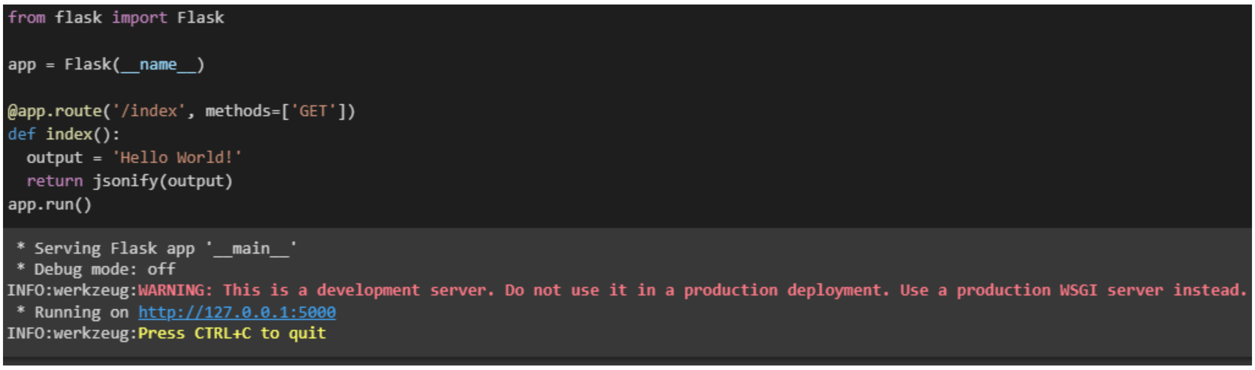
- Flask may not be the best choice for building large or complex APIs, as it lacks some of the more advanced features and functionality of other frameworks.
- Flask does not provide built-in support for authentication and authorization, which may need to be implemented separately.
- Flask is a micro-framework, which means that it requires more manual configuration and setup than some other frameworks.

Install flask version 2.2.3



```
!pip install flask==2.2.3

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: flask==2.2.3 in /usr/local/lib/python3.9/dist-packages (2.2.3)
Requirement already satisfied: importlib-metadata>=3.6.0 in /usr/local/lib/python3.9/dist-packages (from flask==2.2.3) (6.3.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /usr/local/lib/python3.9/dist-packages (from flask==2.2.3) (2.2.3)
Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.9/dist-packages (from flask==2.2.3) (3.1.2)
Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.9/dist-packages (from flask==2.2.3) (8.1.3)
Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.9/dist-packages (from flask==2.2.3) (2.1.2)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.9/dist-packages (from importlib-metadata>=3.6.0->flask==2.2.3) (3.15.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from Jinja2>=3.0->flask==2.2.3) (2.1.2)
```

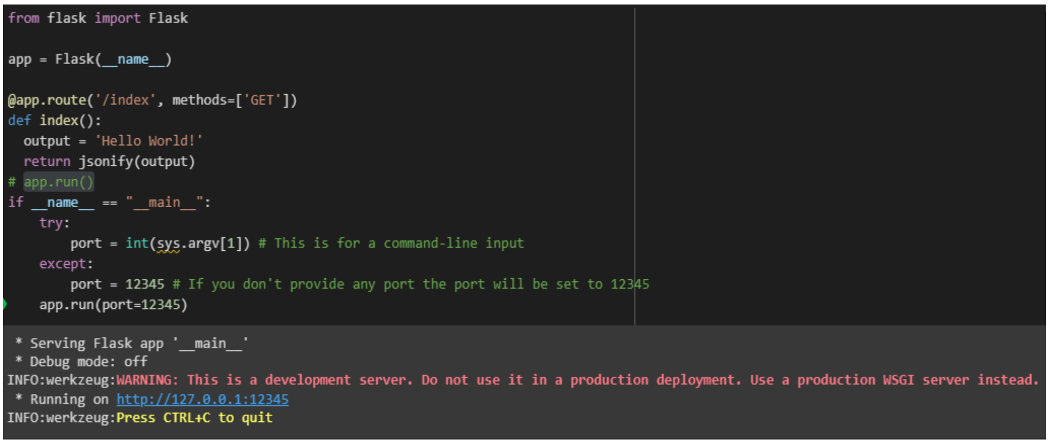
Run flask by running the python file, basically if you won’t specify any port flask run’s on port number 5000.



```
from flask import Flask

app = Flask(__name__)

@app.route('/index', methods=['GET'])
def index():
  output = 'Hello World!'
  return jsonify(output)
app.run()
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
```

If you specify a particular port it runs on that specified port number. Here it's running on port number 12345.

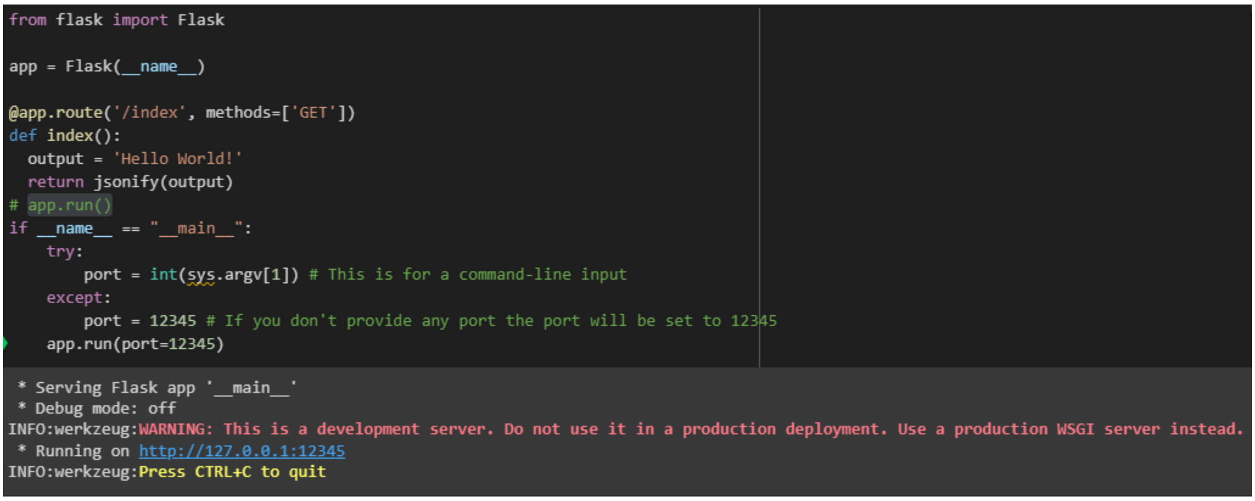


```
from flask import Flask

app = Flask(__name__)

@app.route('/index', methods=['GET'])
def index():
  output = 'Hello World!'
  return jsonify(output)
# app.run()
if __name__ == "__main__":
    try:
        port = int(sys.argv[1]) # This is for a command-line input
    except:
        port = 12345 # If you don't provide any port the port will be set to 12345
    app.run(port=12345)
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:12345
INFO:werkzeug:Press CTRL+C to quit
```

The URL generated by Flask for an API endpoint will depend on the specific route that is defined in the Flask application.

In Flask, we can define a route using a decorator such as @app.route('/my-endpoint'), where /my-endpoint is the path for the API endpoint. For example:



```
from flask import Flask

app = Flask(__name__)

@app.route('/index', methods=['GET'])
def index():
  output = 'Hello World!'
  return jsonify(output)
# app.run()
if __name__ == "__main__":
    try:
        port = int(sys.argv[1]) # This is for a command-line input
    except:
        port = 12345 # If you don't provide any port the port will be set to 12345
    app.run(port=12345)
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:12345
INFO:werkzeug:Press CTRL+C to quit
```

In this example, the route /index is defined with the @app.route decorator, and the function index returns the string 'Hello, World!' when this endpoint is accessed.

The URL for this API endpoint takes the form http://<hostname>/<route>, where <hostname> is the hostname of the server running the Flask application i.e 127.0.0.1 (or) 192.168.34.133, and <route> is the path specified in the @app.route decorator i.e /index here.

So in this case, the URL for the index endpoint would be http://127.0.0.1:12345/index (or) http://198.168.34.133:12345/index, where <hostname> is the hostname of the server running the Flask application. This URL could be used to access the API and retrieve the response generated by the index() function.

Different types of requests to API generated by flask:

Flask allows you to handle different types of HTTP requests, including:

GET - Used to retrieve data from the server. This is the most commonly used request type for retrieving resources from a web API.

POST - Used to submit data to be processed by the server, typically used to create a new resource on the server.

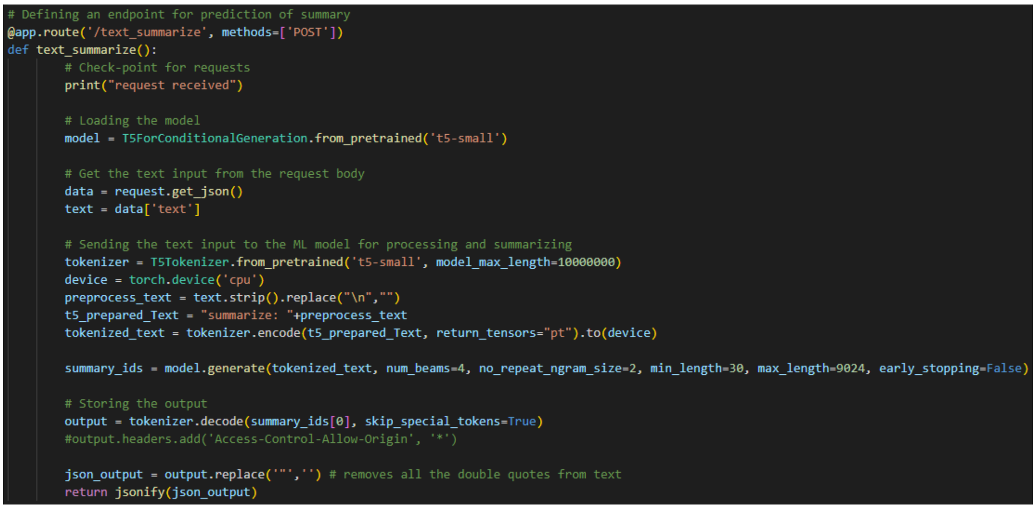
PUT - Used to update an existing resource on the server.

DELETE - Used to delete an existing resource on the server.

But we are only using POST method which would also do the work of GET method if necessary.

POST → used to send some data and retrieve data from the server.

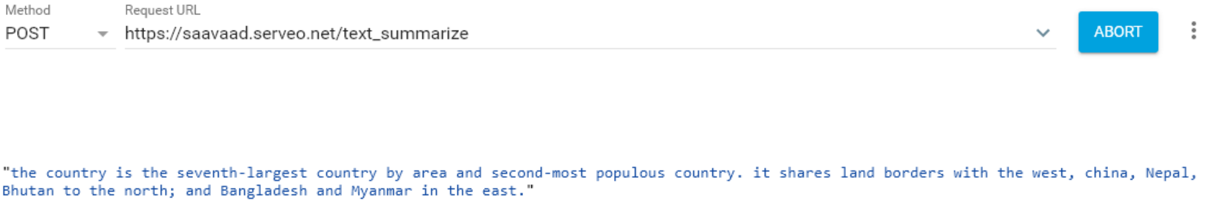
POST Request :



```
# Defining an endpoint for prediction of summary
@app.route('/text_summarize', methods=['POST'])
def text_summarize():
        # Check-point for requests
        print("request received")

        # Loading the model
        model = T5ForConditionalGeneration.from_pretrained('t5-small')

        # Get the text input from the request body
        data = request.get_json()
        text = data['text']

        # Sending the text input to the ML model for processing and summarizing
        tokenizer = T5Tokenizer.from_pretrained('t5-small', model_max_length=10000000)
        device = torch.device('cpu')
        preprocess_text = text.strip().replace("\n","")
        t5_prepared_Text = "summarize: "+preprocess_text
        tokenized_text = tokenizer.encode(t5_prepared_Text, return_tensors="pt").to(device)

        summary_ids = model.generate(tokenized_text, num_beams=4, no_repeat_ngram_size=2, min_length=30, max_length=9024, early_stopping=False)

        # Storing the output
        output = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
        #output.headers.add('Access-Control-Allow-Origin', '*')

        json_output = output.replace('"','') # removes all the double quotes from text
        return jsonify(json_output)
```

POST Response :

Sending some input to the server and receiving the response sent from server.



Method: POST

Request URL: https://saavaad.serveo.net/text_summarize

```
"the country is the seventh-largest country by area and second-most populous country. it shares land borders with the west, china, Nepal, and Bhutan to the north; and Bangladesh and Myanmar in the east."
```

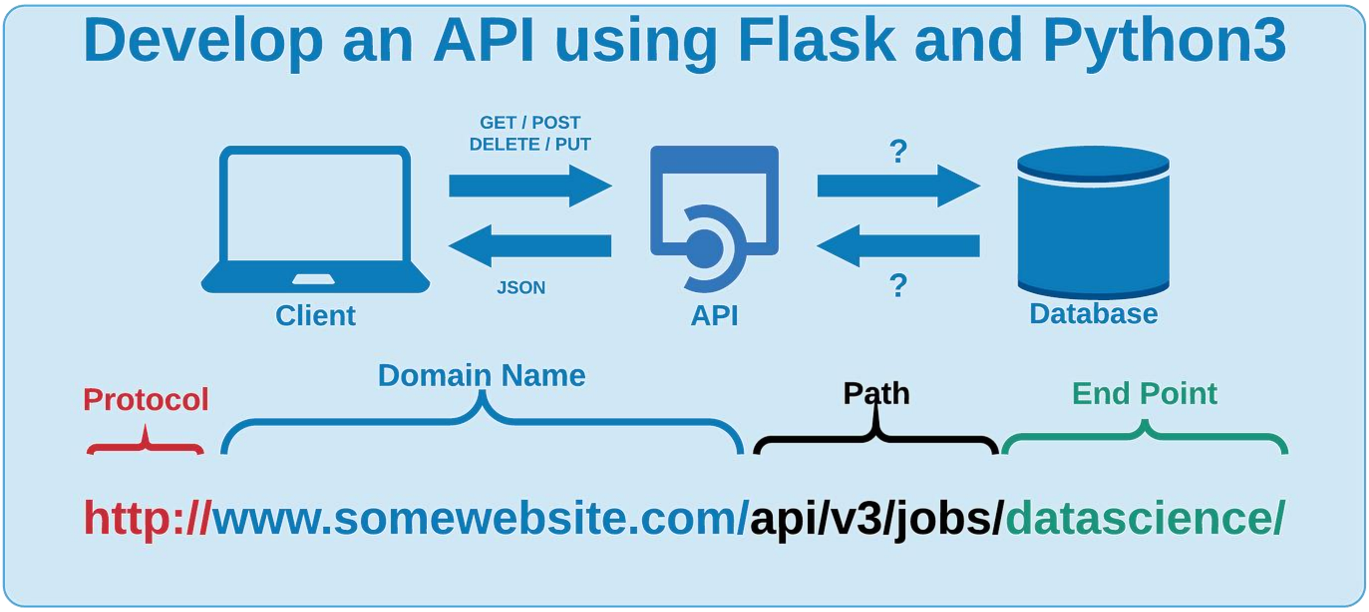
Connections between the Server and the Client through the flask API URL :

When a client (such as a web browser or a mobile app) makes a request to a Flask API endpoint, the request is sent over the internet as an HTTP request. The Flask API server receives the request, processes it, and sends back an HTTP response to the client.

Here is a basic overview of how the connection is made:

- The client makes a request to the Flask API server using a URL.
- The Flask API server receives the request and processes it according to the endpoint that was specified in the URL.
- The Flask API server then sends a response back to the client, usually in the form of JSON data or a rendered HTML page.
- The client receives the response and can then display the data to the user or use it to update the user interface.

Here is a visual representation of this process:


Develop an API using Flask and Python3
GET / POST
DELETE / PUT
JSON
?
?
Client
API
Database
Protocol
Domain Name
Path
End Point
http://www.somewebsite.com/api/v3/jobs/datascience/

Flask- CORS :

Flask-CORS is a Flask extension for handling Cross-Origin Resource Sharing (CORS), which is a security feature implemented by web browsers that restricts web pages from making requests to a different domain than the one that served the original web page. CORS can be a problem when building web applications that need to make cross-domain requests to APIs, for example.

Flask-CORS allows developers to configure the Cross-Origin Resource Sharing headers for their Flask applications, which can help enable cross-domain requests from browsers. It is used to allow the server to specify who can access its resources, what methods are allowed, and what headers can be used. In short, Flask-CORS makes it easier to build web applications that communicate with other web services or APIs.

4.3.5 GLOBAL_API :

What is Global API?

"Global API" is a term that generally refers to an API that is accessible from anywhere on the internet. It means that the API is hosted on a server that is publicly available and can be accessed by clients from different locations, platforms, and devices.

- A global API is useful for applications that need to serve clients from different geographical locations and platforms.

- In order to make an API global, it needs to be hosted on a server that is accessible from the internet. This can be achieved by deploying the API on a cloud-based hosting platform or on a dedicated server that has a public IP address. Once the API is hosted, clients can access it by making HTTP requests to its endpoints using a unique URL or domain name.
- For example, a social media platform needs to provide its services to users from different parts of the world, and a global API allows it to do so.

Different ways of making API Global :

There are different ways to make an API global, the below mentioned ways could help in making Global API:

Self-hosting on a dedicated server: This involves setting up a server with a public IP address and hosting the API on it. This gives full control over the server and the API, but requires technical knowledge and can be expensive.

Cloud-based hosting: This involves deploying the API on a cloud-based hosting platform, such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP). This is more scalable and cost-effective than self-hosting, but requires knowledge of cloud computing platforms.

API gateway: This involves using an API gateway service, such as Amazon API Gateway or Apigee, to manage and secure the API. This is a convenient way to deploy, manage, and secure APIs, but can add extra cost and complexity.

Tunneling services: This involves using a tunneling service, such as ngrok or Serveo, to create a secure tunnel between the API server and the client. This is a quick and easy way to make an API globally accessible, but may not be as secure or scalable as other methods.

Why SERVEO.net and about it:

There are many ways of making API global. Listing few of them here, we have: -

- Ngrok.
- Serveo.net.
- PageKite.
- LocalTunnel.

Everything listed above does the same task, but it is up to the user to make a choice of his own to select one which suits him the best according to his conditions.

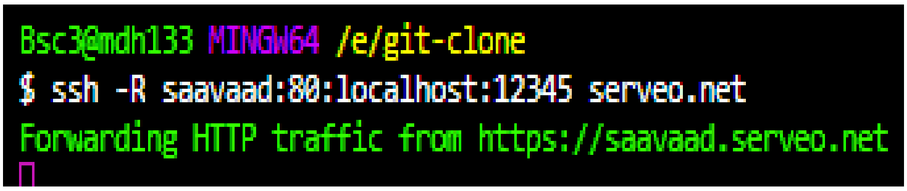
We haven’t selected Ngrok because, that has been blocked by our sophos. And then Serveo.net, which is selected by us because, the primary reason for selecting it is, it’s not being blocked by our sophos, and secondary it’s one among all which is secure compared to others.

Serveo.net is a tunneling service that allows users to expose a local server behind a NAT(Network Address Translation) or firewall to the internet. It works by creating a secure SSH tunnel between the local server and the Serveo.net server, which acts as a proxy for the client. Serveo.net provides a unique URL that clients can use to access the local server from anywhere on the internet. Serveo.net is free to use and doesn't require any installation or configuration.

Security :

To make a Flask API secure through Serveo.net, you can use HTTPS instead of HTTP. HTTPS is a secure protocol that encrypts the data being transmitted between the client and the server, making it more difficult for hackers to intercept and read the data.

Serveo.net provides an option to use HTTPS with a custom domain, which you can set up by following the instructions on their website.



```
Bsc3@mdh133 MINGW64 /e/git-clone
$ ssh -R saavaad:80:localhost:12345 serveo.net
Forwarding HTTP traffic from https://saavaad.serveo.net
```

Serveo.net uses the SSH protocol to create a secure tunnel between the local server and the Serveo.net server.

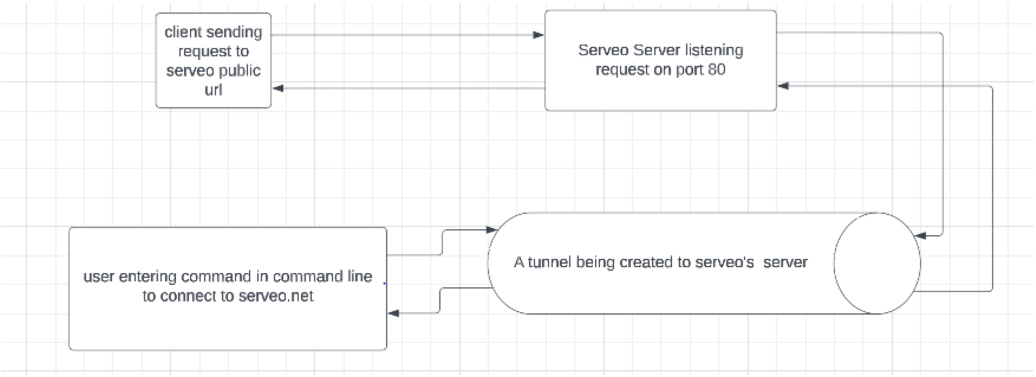
The SSH tunnel is established when the client connects to the unique URL provided by Serveo.net.

Once the tunnel is established, the client can send requests to the Flask API through the Serveo.net server, which acts as a proxy for the local server.

The Flask API processes the requests and sends the response back through the tunnel to the client.

Serveo.net does not store any of the data transmitted through the tunnel, ensuring that the data remains private and secure.

It's important to note that making an API global also comes with certain security considerations, such as protecting against unauthorized access and ensuring data privacy. Therefore, it's important to take necessary security measures while designing, developing, and deploying a global API.

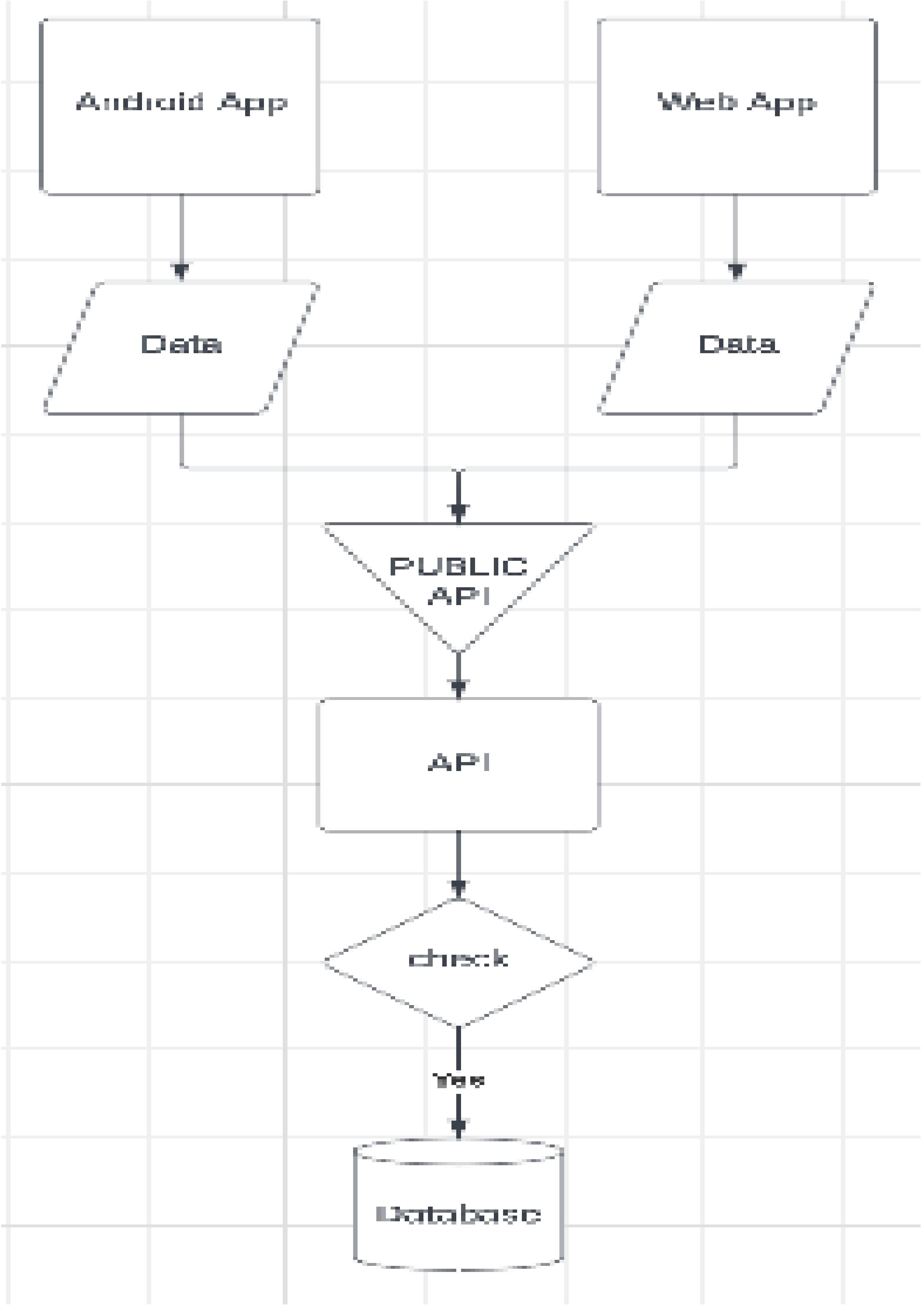

client sending request to serveo public url
Serveo Server listening request on port 80
user entering command in command line to connect to serveo.net
A tunnel being created to serveo's server

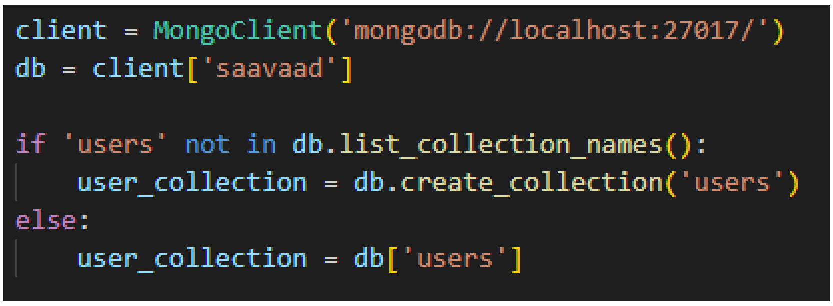
4.4 Database (Mongo) :

Having database for any product is very crucial as we discussed earlier. To help user save all his data and retrieve them whenever necessary, we are using MongoDB storage. As we have two different things i.e android application and web application, both the applications need to use the same database. For this we thought of using MongoDB cluster. That did not work in our local machines.

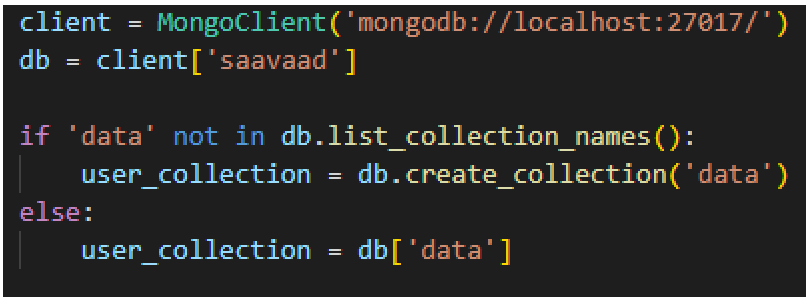
Tackling the situation we came with a new idea thinking why can't we use flask and do something like both the applications could use the same database.

As we have already seen both android and web application are accessing the same Machine Learning Models through API generate by Flask and Serveo. We thought to expose our local database storage through Flask and Serveo, such that both the applications could make use of single storage, the functionality is same as MongoDB cluster, but the way of implementation is changing here.


Android App
Web App
Data
Data
PUBLIC API
API
check
Yes
Database



```python
client = MongoClient('mongodb://localhost:27017/')
db = client['saavaad']

if 'users' not in db.list_collection_names():
    user_collection = db.create_collection('users')
else:
    user_collection = db['users']
```



```python
client = MongoClient('mongodb://localhost:27017/')
db = client['saavaad']

if 'data' not in db.list_collection_names():
    user_collection = db.create_collection('data')
else:
    user_collection = db['data']
```

All the data storage and retrieval is done with a simple yet powerful pymongo library of python. These are functions or features of our database storage system.

Login/ Signin

Signup

Forgot Password

Store Data

Retrieve Data

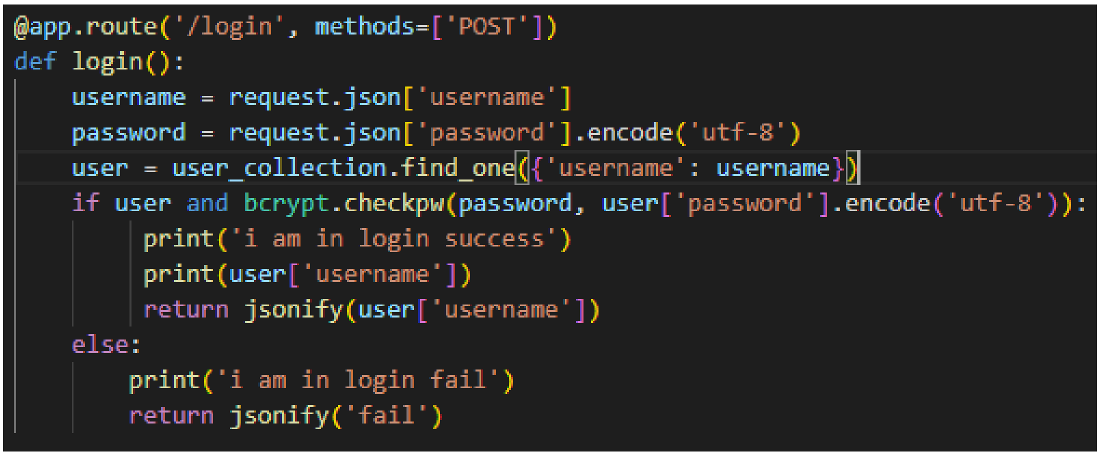
Login/ Signin :

The user sends the details of his/her username and password, those are being received in the flask api end point as json and checks whether a user exists with entered credentials or not, if a user doesn’t exist, it would return ‘fail’ message. If a user with the username exists, then the password is verified with the existing user password.

The key highlights of our storage system is, even the owner of the database never knows the password of the user. Because the password being sent by the user is encrypted using hashing and bcrypt.

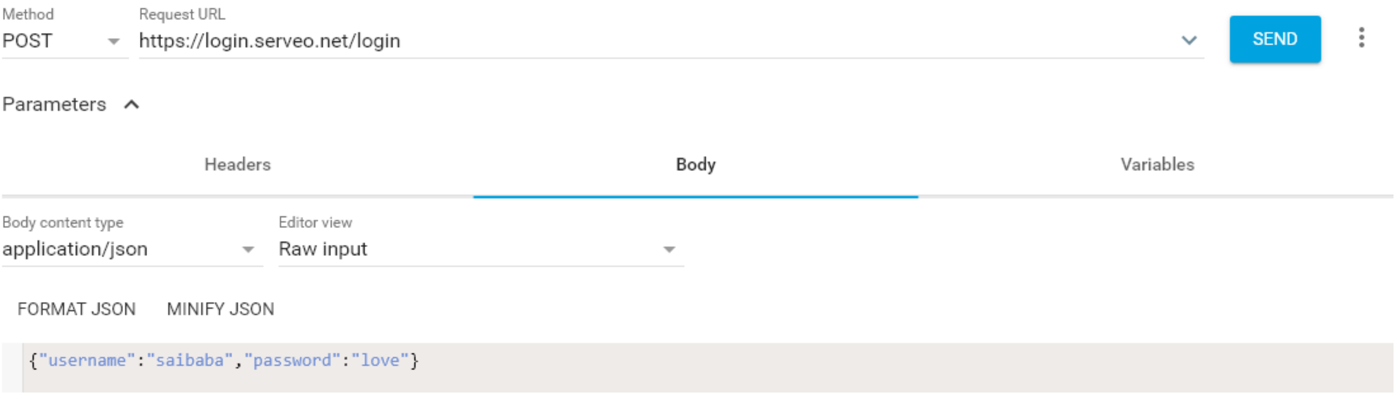
If the login is successful, then the API returns the username to the user.

API end point for login :



```python
@app.route('/login', methods=['POST'])
def login():
    username = request.json['username']
    password = request.json['password'].encode('utf-8')
    user = user_collection.find_one({'username': username})
    if user and bcrypt.checkpw(password, user['password'].encode('utf-8')):
        print('i am in login success')
        print(user['username'])
        return jsonify(user['username'])
    else:
        print('i am in login fail')
        return jsonify('fail')
```

Output :



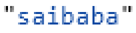
Method: POST

Request URL: https://login.serveo.net/login

Body content type: application/json

Editor view: Raw input

```
{"username":"saibaba","password":"love"}
```



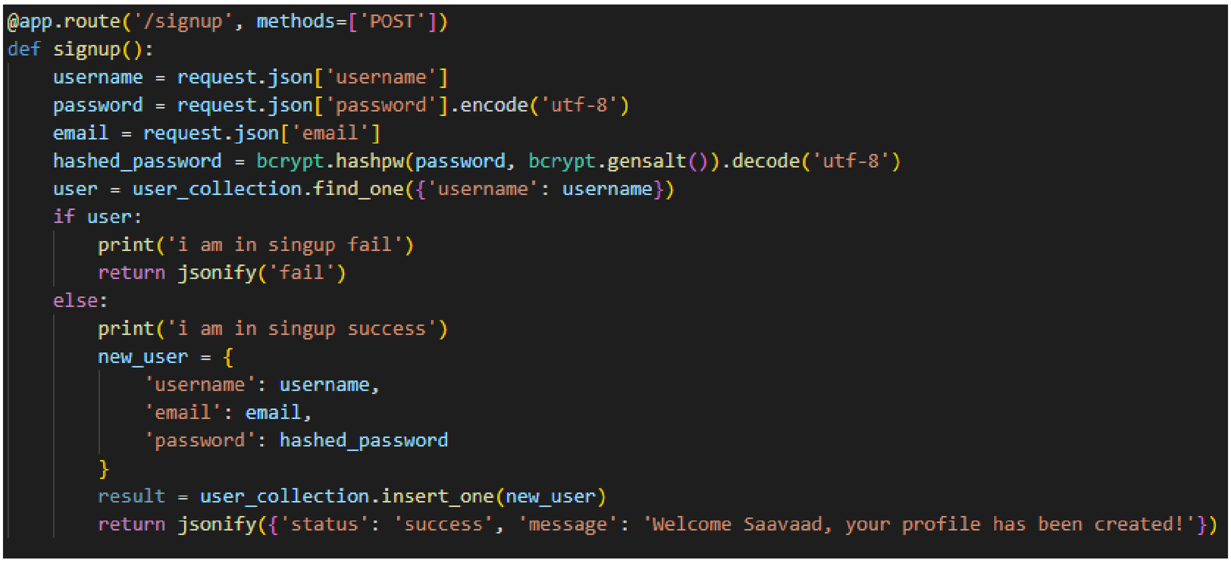
```
"saibaba"
```

--- username being returned from the API after login.

Signup :

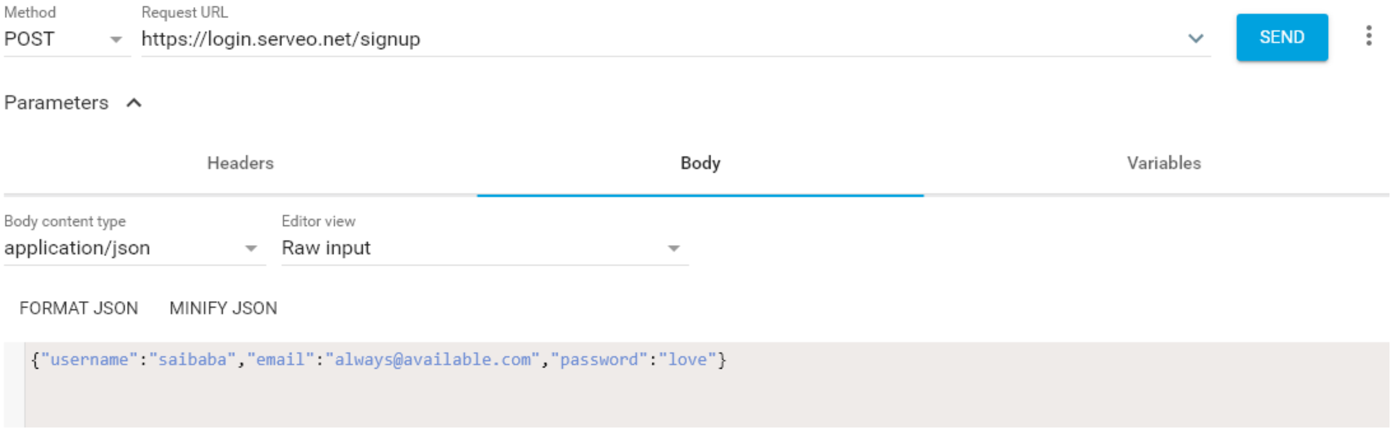
If a user doesn’t have an account he can create his account by going to the signup page and enter the details such as username, email, password, if a user with an entered username exists, then new account could not created and a prompt to user would be given to retry with different username, if no account with entered username exists, then user could successfully finish his signup and would redirect to login page, the password encryption and decryption also exists here.

API end point for signup



```
@app.route('/signup', methods=['POST'])
def signup():
    username = request.json['username']
    password = request.json['password'].encode('utf-8')
    email = request.json['email']
    hashed_password = bcrypt.hashpw(password, bcrypt.gensalt()).decode('utf-8')
    user = user_collection.find_one({'username': username})
    if user:
        print('i am in singup fail')
        return jsonify('fail')
    else:
        print('i am in singup success')
        new_user = {
            'username': username,
            'email': email,
            'password': hashed_password
        }
        result = user_collection.insert_one(new_user)
        return jsonify({'status': 'success', 'message': 'Welcome Saavaad, your profile has been created!'})
```

Output :



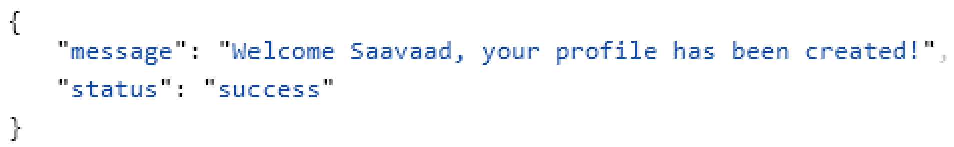
Method: POST

Request URL: https://login.serveo.net/signup

Body content type: application/json

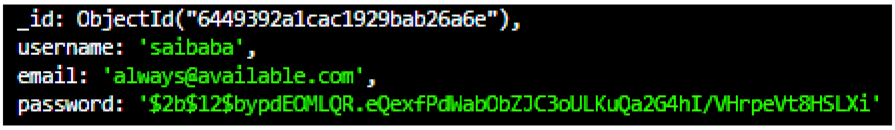
Editor view: Raw input

```
{"username":"saibaba","email":"always@available.com","password":"love"}
```



```
{
    "message": "Welcome Saavaad, your profile has been created!",
    "status": "success"
}
```

Data stored in database

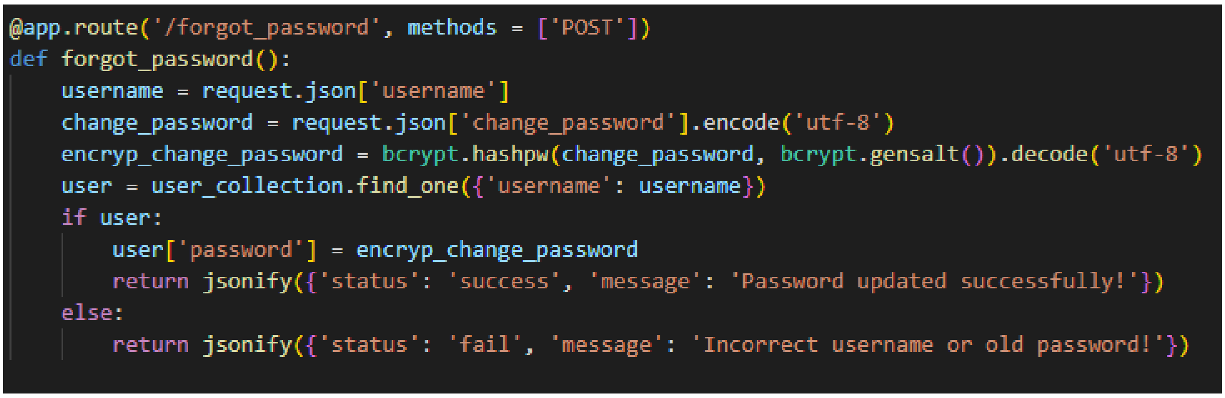


```
_id: ObjectId("6449392a1cac1929bab26a6e"),
username: 'saibaba',
email: 'always@available.com',
password: '$2b$12$bypdEOMLQR.eQexfPdWabObZJC3oULKuQa2G4hI/VHrpeVt8HSLXi'
```

Forgot Password :

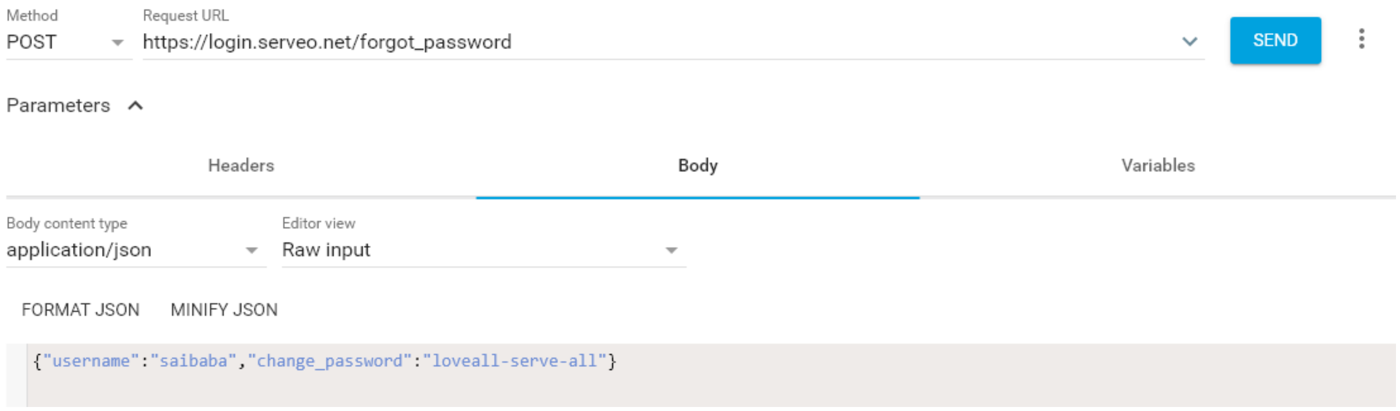
Not much authencation is added, but a feature to change or update or forgot password is provided, if the user forgets his password, the user enter his username and enter a new password in the forgot password page, if a user with entered username exists, then his password would be updated with the new password entered in hat page, he could redirect to login page and try with the new password. If a user doesn't with the entered username, user needs to signup to create an account.

API end point for forgot password :



```
@app.route('/forgot_password', methods = ['POST'])
def forgot_password():
    username = request.json['username']
    change_password = request.json['change_password'].encode('utf-8')
    encryp_change_password = bcrypt.hashpw(change_password, bcrypt.gensalt()).decode('utf-8')
    user = user_collection.find_one({'username': username})
    if user:
        user['password'] = encryp_change_password
        return jsonify({'status': 'success', 'message': 'Password updated successfully!'})
    else:
        return jsonify({'status': 'fail', 'message': 'Incorrect username or old password!'})
```

Output



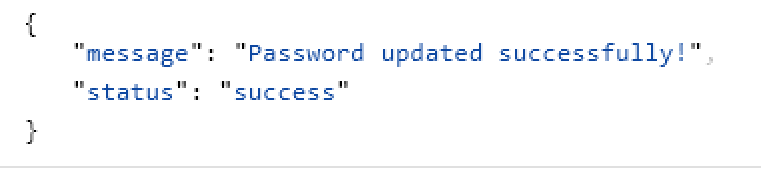
Method: POST

Request URL: https://login.serveo.net/forgot_password

Body content type: application/json

Editor view: Raw input

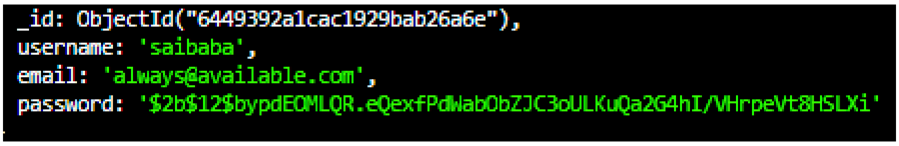
```
{"username":"saibaba","change_password":"loveall-serve-all"}
```



```
{
    "message": "Password updated successfully!",
    "status": "success"
}
```

Change of password in Database

Updated password



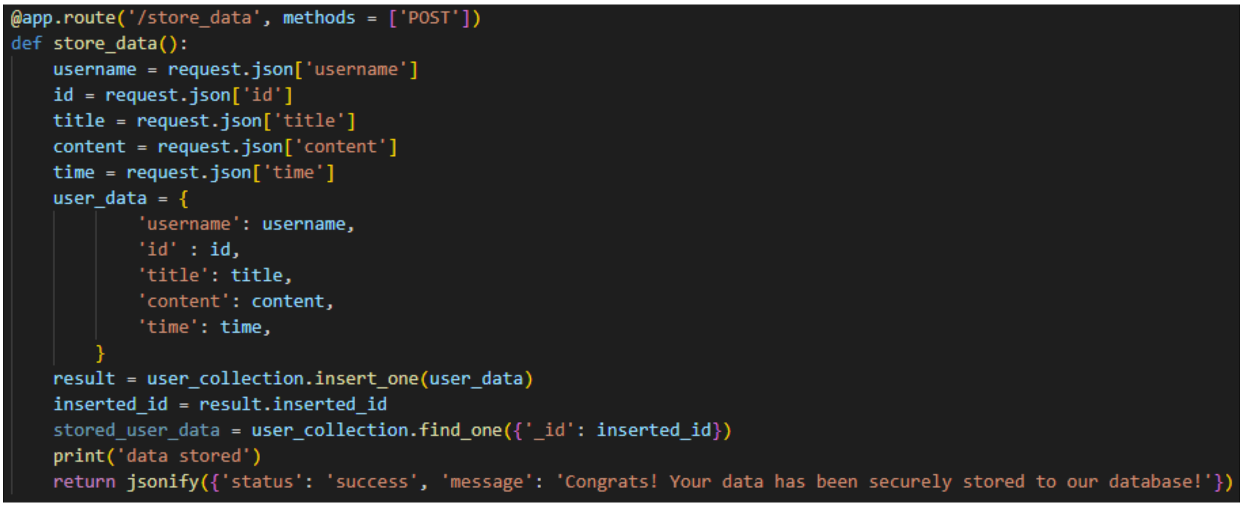
```
_id: ObjectId("6449392a1cac1929bab26a6e"),
username: 'saibaba',
email: 'always@available.com',
password: '$2b$12$bypdEOMLQR.eQexfPdWabObZJC3oULKuQa2G4hI/VHrpeVt8HSLXi'
```

Store Data :

Though a hierarchial data storage hasn’t been maintained, but a decent enough data storage to store all the data being generated by the user is stored in our database.

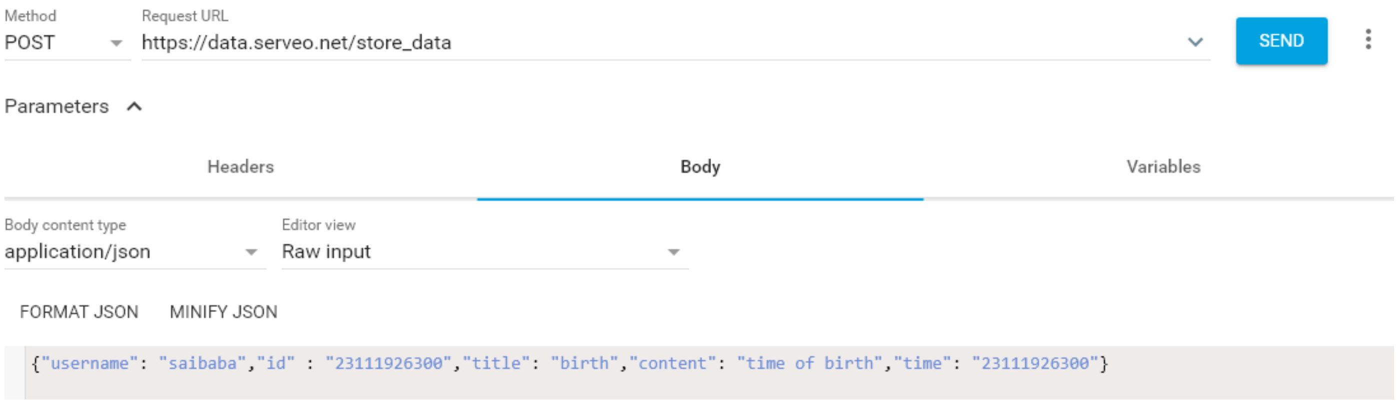
If a user performs any action like summarization or translation, the text output being generated by the application could be stored with the title given by user, along with username, time created, content and a unique id for each and every document created.

API end-point to store the data from the application



```
@app.route('/store_data', methods = ['POST'])
def store_data():
    username = request.json['username']
    id = request.json['id']
    title = request.json['title']
    content = request.json['content']
    time = request.json['time']
    user_data = {
            'username': username,
            'id' : id,
            'title': title,
            'content': content,
            'time': time,
        }
    result = user_collection.insert_one(user_data)
    inserted_id = result.inserted_id
    stored_user_data = user_collection.find_one({'_id': inserted_id})
    print('data stored')
    return jsonify({'status': 'success', 'message': 'Congrats! Your data has been securely stored to our database!'})
```

Output



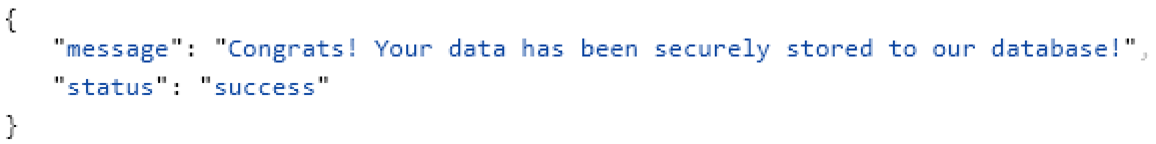
Method: POST

Request URL: https://data.serveo.net/store_data

Body content type: application/json

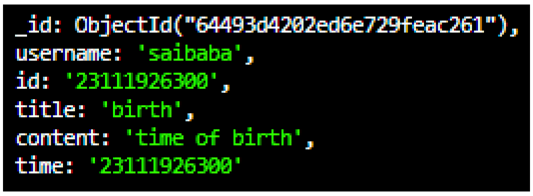
Editor view: Raw input

```
{"username": "saibaba","id" : "23111926300","title": "birth","content": "time of birth","time": "23111926300"}
```



```
{
    "message": "Congrats! Your data has been securely stored to our database!",
    "status": "success"
}
```

Data stored in Database



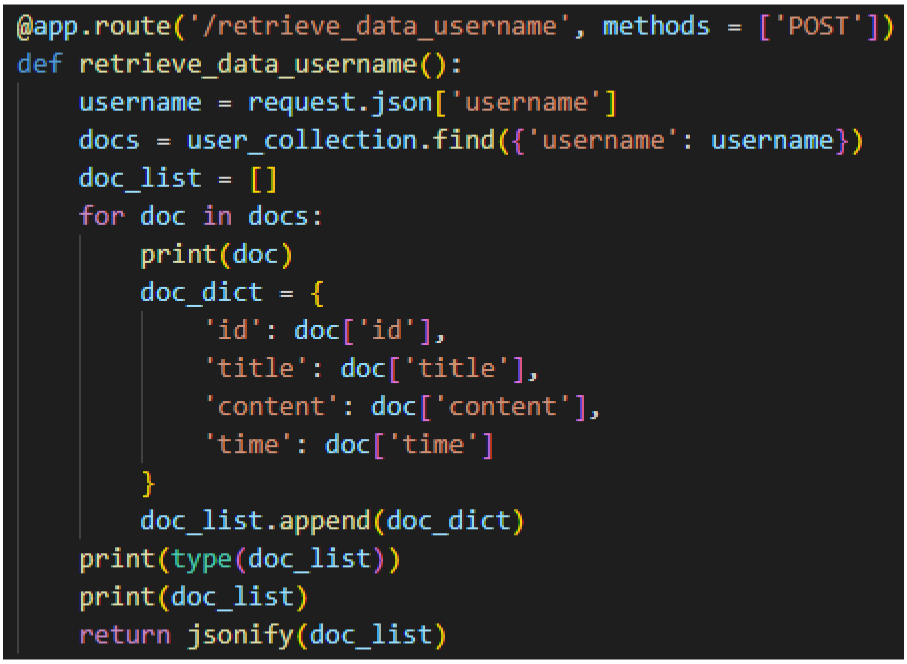
```
_id: ObjectId("64493d4202ed6e729feac261"),
username: 'saibaba',
id: '23111926300',
title: 'birth',
content: 'time of birth',
time: '23111926300'
```

Retrieve Data :

Though a hiearchial data storage hasn't been maintained, but a decent enough data storage to store all the data being generated by the user is stored in our database.

Once the user enters into our website or android application, we could display all of his collections. For this we need to retrieve data that is being stored under the username.

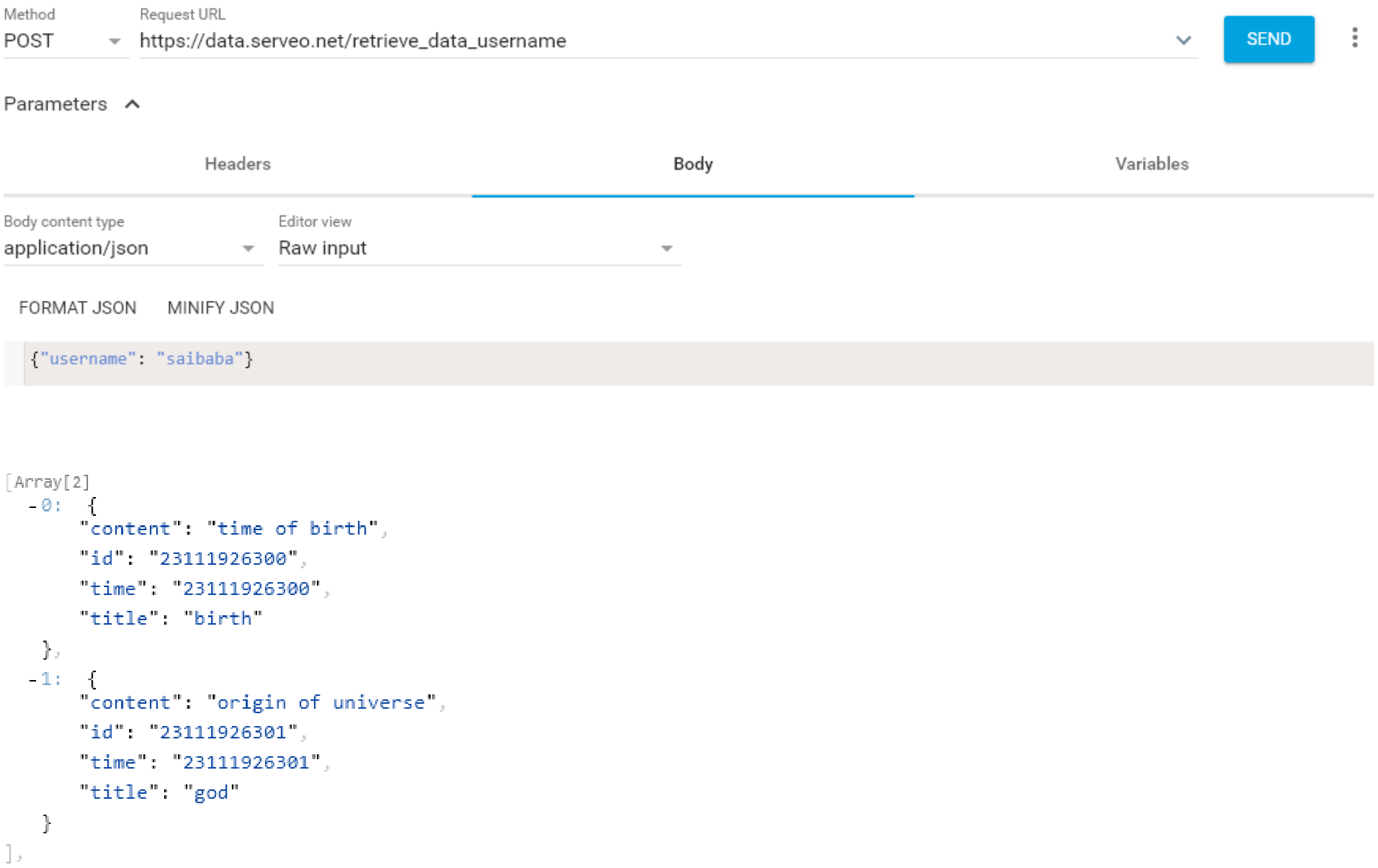
The username is being sent by the application to the API and all the data related to entered username is put up in array and a json object output would be returned from the API end point, which is then received by the application and retrieve all the data from array and display them.



```
@app.route('/retrieve_data_username', methods = ['POST'])
def retrieve_data_username():
    username = request.json['username']
    docs = user_collection.find({'username': username})
    doc_list = []
    for doc in docs:
        print(doc)
        doc_dict = {
            'id': doc['id'],
            'title': doc['title'],
            'content': doc['content'],
            'time': doc['time']
        }
        doc_list.append(doc_dict)
    print(type(doc_list))
    print(doc_list)
    return jsonify(doc_list)
```

API end point to retrieve data of the user

Output



Method: POST

Request URL: https://data.serveo.net/retrieve_data_username

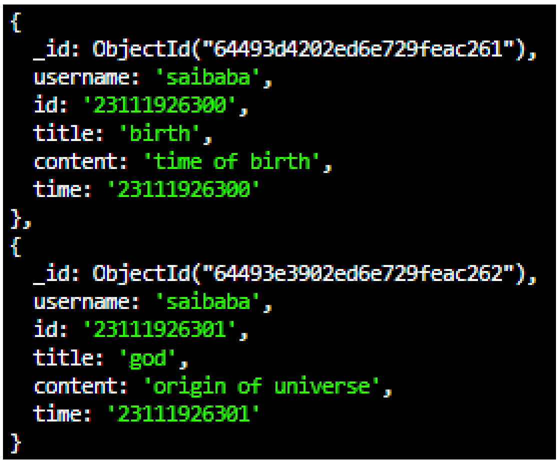
Body content type: application/json

Editor view: Raw input

```
{"username": "saibaba"}
```

```
[Array[2]
  -0:  {
       "content": "time of birth",
       "id": "23111926300",
       "time": "23111926300",
       "title": "birth"
   },
  -1:  {
       "content": "origin of universe",
       "id": "23111926301",
       "time": "23111926301",
       "title": "god"
   }
],
```

Data from Database



```
{
  _id: ObjectId("64493d4202ed6e729feac261"),
  username: 'saibaba',
  id: '23111926300',
  title: 'birth',
  content: 'time of birth',
  time: '23111926300'
},
{
  _id: ObjectId("64493e3902ed6e729feac262"),
  username: 'saibaba',
  id: '23111926301',
  title: 'god',
  content: 'origin of universe',
  time: '23111926301'
}
```

CHAPTER 5
CONCLUSION & SCOPE

Conclusion :

With the current pace at what we are developing, the need for development may reduce in future, but the need to utilise the resources already available to make a new resource is essential, enough researches and developments are undergoing to innovate new things, the time for integration of innovation and intelligence has come, hence there exists a great demand for artificial intelligence and machine learning concepts.

A project with core topics related to them might help the student understand about it very well and would give the boost for studying such courses.

At the end of the day what we expect from the world is a sustainable development, a user and eco friendly environment to sustain in this world. Our application provides the best environment for the user to make the best use of it.

The main objective of this project is to provide a feasible and a free product to user where each and every user with basic requirements of internet and minimal resources could use it with simple understanding about the technology.

This product could be used by anyone and especially students where if they find reading a big book or an essay might be difficult, for them this would be very handy to use and finish their task happily.

Future scope :

This project isn’t limited for a short period, it’s has a huge future scope of developing it, more and more features could be integrated to this project to provide wide variety of features to the user.

More and more sophisticated machine learning models can be used to apprehend the accuracy of the output, more secured database services of storage and retrieval of data could be implemented.

The project could be improvised to take the input as a speech and then making the summary for the given speech.

Many other languages could be added for translation and large models be used for good accuracy depending on the processors and systems where the summarization and translation takes place.

CHAPTER 6
BIBILOGRAPHY

Transformers – Hugging face community --- https://huggingface.co/

Kraken --- https://kraken.re/main/index.html

ChatGPT --- https://openai.com/blog/chatgpt

Pytesseract --- https://pypi.org/project/pytesseract/

Flask --- https://flask.palletsprojects.com/en/2.3.x/

Serveo.net --- http://serveo.net/

MongoDB --- https://www.mongodb.com/