Implementation Of An Asynchronous Document Embedding Uploader API

**Proposal for C4GT 2023 | Project Idea: Document Uploader**

*SanchariRay*

**Sanchari Ray | Date: 10th June 2023 |**

## Synopsis

In today's technological environment, **AI tools** have become crucial. These tools make use of machine learning and artificial intelligence to improve decision-making, automate tasks, and draw conclusions from data. AI technologies provide a wide range of functions that streamline complex operations and spur innovation across industries, from **natural language processing** to computer vision. Machines are now capable of comprehending, analysing, and producing human language thanks to Natural Language Processing (NLP) techniques. They make **text classification**, **sentiment analysis**, and **language translation** easier.

The proposed idea leverages **Python, CLI, Natural Language Processing (NLP), System Design, Data Hugging Transformers , FastAPI, UVicorn** technologies to enable the development of an asynchronous Document Embedding Uploader API. AsyncDocEmbed is a cutting-edge project that seeks to transform document embedding and sharing. It offers a simple API for uploading files, creating asynchronous embeddings for document chunks, and allowing users to download the entire file whenever they want. Hugging Face Transformers and NLP approaches are combined to give AsyncDocEmbed users the ability to extract valuable representations from their documents and apply them to a variety of applications. Proposed implementation details and **Scalable High-Level system Designs (HLDs)** have been discussed in the proposal.

The asynchronous nature of uploading and processing document chunks underlies the project's main functionality. The API allows users to quickly upload document chunks, and safe temporary storage of the data is provided. The Hugging Face Transformers library is then used to process the uploaded chunks using cutting-edge NLP techniques like **tokenization** and **embedding creation**. By capturing the semantic meaning and context of the document chunks, these embeddings enable users to get insightful information and carry out sophisticated research.

***Keywords:*** *Asynchronous, Document Uploader, API, NLP, Scalable*

# Relevant Skills And Motivation

## Languages of Choice and Relation to the project

**Backend:** The preferred language has been Python. The majority of the proposed implementations use **FastAPI**, a high-performance web framework for creating Python APIs, although the most of the proposal's components entail explicitly coding in Python. Regarding our current use case, we suggest using **UVicorn** and **Hugging Face Transformers**. UVicorn is incredibly effective and supports asynchronous execution, making it the perfect choice for serving the API. Hugging Face Transformers offers a wide range of pre-trained models that can be used to process document chunks and generate embeddings. Despite the availability of alternatives like Docker and Kubernetes, FastAPI and UVicorn were chosen instead for the following 2 key reasons:

* **Familiarity:** Utilising FastAPI, Hugging Face Transformer, and UVicorn for one's own projects resulted in a shorter development time because of improved familiarity.
* **Close Modelling of Proposed Scalable System:** This point would be easier to understand while going through the proposed High-Level Design of a scalable microservice architecture. FastAPI has the capability of leverages the power of asynchronous programming, allowing for high-performance and scalable API development. FastAPI seamlessly integrates with asynchronous libraries like Hugging Face Transformers and UVicorn, allowing for efficient and concurrent processing of document chunks whereas Hugging Face Transformers provides access to a vast collection of pre-trained NLP models, including popular models like **BERT, GPT**, and more.

It's also important to keep in mind that UVicorn, an ASGI server, provides the backend infrastructure needed to run FastAPI programmes. In order to process requests and produce responses, it manages incoming HTTP requests, concurrent connections, and communication with the FastAPI framework. Through its built-in **asyncio** module, Python also comes with built-in functionality for handling asynchronous requests.

### Resources to get an estimate of speed, performance testing and costs involved

Performance testing of API can be conducted using tools like **Apache JMeter, Locust**, or **Siege**. These tools can simulate concurrent requests and measure the response times and throughput of the API. By analyzing the results, we can get an idea of the speed and scalability of the API. Pricing calculators are provided by cloud service providers including Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure. These tools let us calculate the expenses of maintaining an API based on variables like the volume of requests, the availability of computational resources, storage space, and data transfer. We can obtain an estimate of the monthly prices by entering our anticipated consumption metrics.

The intricacy of your API logic, the size of your document chunks, the number of concurrent requests, the particular cloud provider and services people use, and other factors may all affect the real speed and prices.

## Motivation

## Hugging Face Transformers, Python, and NLP all have enormous potential and revolutionary qualities, which is why these technologies were chosen for this project. As a discipline, NLP has transformed how we interact with and comprehend human language, enabling a wide range of applications across numerous industries. With its ease of use, adaptability, and robust ecosystem of libraries, Python offers a strong basis for creating strong and effective solutions. Hugging Face Transformers, a state-of-the-art NLP task library, provides pre-trained models and tools that speed up development and enable us to create complex language processing systems. Utilising NLP's ability to glean insights from unstructured text data will allow us to address challenging linguistic problems and develop creative solutions. The synergy between NLP, Python, and Hugging Face Transformers inspires us to explore new possibilities, push the boundaries of what's achievable, and make a meaningful impact in the world of natural language understanding and processing.

# Project Details

**Proposal:** Implementation Of An Asynchronous Document Embedding Uploader API

**Based on C4GT 2023 Project Idea:** Document Uploader

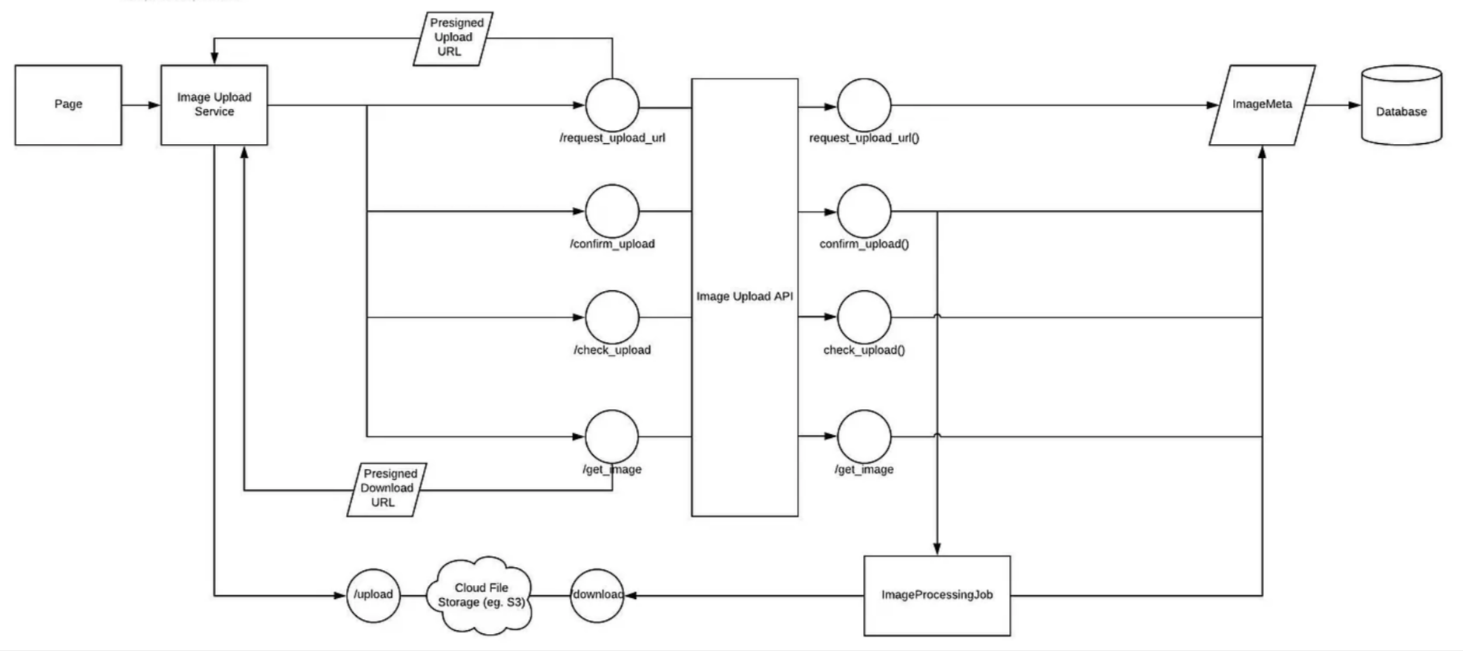
## Project Overview

We can now move on to the actual implementation and the specifics of the various features that this platform shall make available to the stakeholders/entities in question because we have a good enough understanding of the technologies and frameworks that must work together to develop the system under consideration.

### Entities Involved and Privileges Provided

It is crucial to first and foremost have a solid grasp of the numerous stakeholders (also referred to as entities in this proposal) engaged in the system before launching right into the development of the proposed system and its features. Formally, we describe an entity/stakeholder as a person or group that has the necessary rights to access the private and secure API and see or edit its contents.

Kindly refer to the diagram given below and the detailed privileges provided to each of the entities involved in the proposed system:



***Figma:*** [***https://www.figma.com/community/file/1249279969049992273***](https://www.figma.com/community/file/1249279969049992273)

Entities involved and their respective powers/privileges:

1. **Docs/Image Upload API:** The lifespan of an Docs upload request is managed by the Docs upload API. It will handle things like authentication, authorisation, auditing, rate limitation, and other things, but it will let other services handle the actual administration of the file data.File data is never altered by it
2. **Docs/Image Upload Service:** The Docs Upload Service hides behind a single method interface the logic required to contact the Docs Upload API endpoints with the correct data in the correct order. It is crucial to have a single source of truth for this method since you don't want this temporal coupling to be distributed throughout your entire system.
3. **Signed URLs:** A URL with permission parameters is known as a signed URL. When uploading and downloading files, we can employ signed URLs to provide protection and guarantee that only authorised users carry out these tasks.

**Signed Upload URLs:** An unsigned URL, such as **/uploads**, might be returned by the simple case of **/docs/get\_upload**\_url.This implies that anyone might just call /uploads and add arbitrary files to our data storage or use it as their own personal file server. This is obviously not what we want.A signed URL could be returned by the command **/docs/get\_upload\_url: /uploads?write\_token=a99Xioajksf23**.This indicates that we have the chance to verify the user's identity, grant them permission, set a rate limit, or check who created the signed URL. We have the tools to stop a user if they abuse our upload service.

**Signed Download URLs:** Anyone who knew the URL **/docs/joseph-gefroh.docx** could download the document if we didn't have signed download URLs.In many situations, this might be a good thing, but in others, such with confidential files, it might be very bad. We wouldn't want the URL to be made public under these circumstances.Images**/get\_download\_url**, which provides a temporary token like: **/downloads?read\_token=a99Xioajksf23**, allows us to block access behind our own endpoint.

1. **Cloud File Storage:** To ease the strain on our web servers, we employ a different file storage provider (like S3).
2. **DocumentProcessingJob :**You don't want to do file processing on our application or web server, which is busy serving other requests, as it might be very memory and CPU intensive.We convert this into an asynchronous call and offload it to another service in order to maintain an unblocked, effective primary web server.

## Setup and Working

**The Algorithm**

Let's look at how we would employ our system's elements to actually upload the document:

* Step 1: The client requests a server's upload URL (REQUEST).
* Step 2: Client uploads the image data to the upload URL (UPLOAD)
* Step 3: Client tells the server the upload is completed (CONFIRM)
* Step 4: Server processes the file in background (PROCESS)
* Step 5: Client checks the file processing status (CHECK)
* Step 6: Server is done processing image, notifies client (FINALIZE)

**Step 1: The client requests a server's upload URL (REQUEST).**

Why isn't a file upload the first step? Because we must keep in mind that we don't want the file data to ever reach our servers. Too many resources are used by it, and it has a denial-of-service vulnerability.

Instead, we request from our server the URL to which we should upload. This URL could lead to a system designed to handle the load of file uploads, such as S3, or to a third-party cloud storage.

During this step, the server can generate a random URL that is:

1. time restricted
2. audited
3. authorized

These upload URLs have already been signed. In other words, the URL the server gives has all of the authorization information needed to upload to the third-party service in the query parameters.

**Step 2: Client uploads the image data to the upload URL (UPLOAD)**

This process is rather simple. After receiving the Upload URL from the server, the client simply issues a POST request to that URL. After that, that service receives and stores the data.

**Step 3: Client tells the server the upload is completed (CONFIRM)**

With the token the server had earlier returned, the client sends a request to the server informing it that the upload has been completed**.**

**Step 4: Server processes the file in background (PROCESS)**

The server examines the token for validity before looking for that upload request.

It starts a job to process the file without delaying requests to the web server, checking the file's integrity, producing variants, and optimising it.

**Step 5: Client checks the file processing status (CHECK)**

It takes some time to process files, therefore you don't want the client to hold up a request. The client should periodically check to verify if the processing is complete**.**

**Step 6: Server is done processing image, notifies client (FINALIZE)**

The check will eventually succeed, and the server will then return the document URL. The image may now be used by the customer at will.

The document URL here provides the security. If the file is password-protected, the URL to view it would lead to our server, and any requests from clients would have to include a read token, which the server could use to complete authorization as necessary and create a temporary signed URL for that file.

If the document is intended to be seen by the general public, the document URL might make a direct reference to the image rather than using our server at all.

### Feature Implementations

We now proceed to discuss the proposed features of the system.

1. **API Design:** Creating the request/response structures and API endpoints. The settings required for document upload, chunking, and embedding generation should be determined. The API routes that handle these activities should be specified.
2. **DocumentUpload:** Implementing the capability to allow user document uploads. This can be accomplished by accepting the document text as input or by using a system for uploadingfiles**.**
3. **Chunking:** Creating the logic necessary to divide the document into more manageable sections. Based on the demands of our application, we need to choose the proper chunk size. This can be done using any criterion, including word count and character count.
4. **NLP Processing:** Processing each document chunk using Hugging Face Transformers. Appling tokenization and any additional preprocessing processes that are required. For each piece, creating embeddings using pre-trained models. The embeddings should be kept in memory or a short-term storage device.

**5.Embedding Retrieval:** Creating and putting into operation a retrieval endpoint for the created embeddings. To accomplish this, either we will provide each chunk a special identification or describe the range of chunks we want to receive.

**6.Temporary Data Storage:** A temporary data storage system to temporarily store the document chunks is to be used and their corresponding embeddings. We can then request downloads or access the created embeddings within a predetermined window of time.

**7.Download Functionality:** Creating a function that allows us to ask for a download of the edited document or the embedded content. This can be done by producing a downloaded file or by offering a download link.

**8. Error Handling and Validation:** Implementing error-handling and validation techniques to make sure the API can gracefully manage unforeseen circumstances. Verification of user input, deal with errors, and send users clear error messages.

1. **Testing and Optimization:** Testing the functionality carefully to make sure it functions as expected and fulfils performance standards. If necessary, optimization of the code needs to be done to boost responsiveness, memory effectiveness, and overall performance.
2. **Documentation:** Providing thorough API documentation that includes usage examples, input/output parameters, and endpoint definitions. Users can better interact with the API by understanding how to do so.

## Problems

The aforementioned suggested system designs have already taken into account concerns like scalability and security. However, there are still two significant issues that could arise during the installation and maintenance of the suggested system. We will address these issues and their remedies in this section.

1. **Continuous Model Updates:** Models and methods for NLP are developing quickly. It can be difficult to stay on top of new developments, update the models, and incorporate them into the project. The project can be kept current by routinely checking for and incorporating updates from Hugging Face Transformers or other pertinent sources.
2. **Long Term Data Retention:** Important factors to take into account include managing temporary data storage and establishing data preservation standards. The project's long-term survival may depend on decisions on how long to keep temporary data, the implementation of cleanup procedures, and consideration of legal and compliance requirements for data retention.

**Possible Solutions**

We can deal with the above-mentioned problems by either preventing them from occurring or by having tools in place that could help us quickly debug / constantly monitor the status of our system. Let us take a quick look at the proposed solutions to the above-mentioned problems which might occur during development.

#### Addressing the needs for continuous model updates

* + **Stay Updated with Hugging Face Transformers:** Keep up with Hugging Face Transformers' releases and updates. To stay up to date on the newest models, enhancements, and problem fixes, subscribe to their release notes, follow their GitHub repository, or sign up for their community forums. This makes sure you can take advantage of the most recent NLP breakthroughs and update your project appropriately.
  + **Automated Model Updates:** Create automated procedures to check for Hugging Face Transformers changes on a regular basis and to schedule model upgrades accordingly. Setting up automated scripts or **CI/CD pipelines** that get the most recent models, carry out tests, and deploy the new models to your API will help with this. By automating this procedure, you can make sure that your project is constantly taking advantage of the most recent advancements**.**

1. **Addressing Long Term Data Retention:** 
   * **Scheduling Data Purging:** Establish recurring procedures to clear out temporary data that has remained after its allotted retention time. This can entail executing scripts or background processes that automatically detect and delete outdated data from your storage systems. Ensure that data deletion is handled securely and in accordance with any applicable privacy laws.
   * **Anonymization or Pseudonymization:** Consider anonymizing or pseudonymizing the data to safeguard user privacy if it is practical for your use case. You can lessen the privacy risks connected with long-term data retention by deleting or encrypting **personally identifiable information (PII).**

## Implementation Details

We can now define deadlines and milestones for the proposal for the Coding Period of C4GT 2023 because we have a thorough understanding of the entire proposed system.

Three significant milestones that outline or signal the completion of three significant portions of the codebase make up the full coding period. The coding cycle lasts for 42 days (6 weeks), or from June 20 to July 31. Please take note that only the project's most important milestones are highlighted in this section. Please refer to the timeline section of the website for a more thorough explanation of the coding period's deliverables.

* **Requirement Analysis and API Design Phase | Milestone 1 (End of Week 1):** Gathering of detailed requirements for the document uploader and designing the API endpoints request/response structures, and overall architecture of the project.
* **Chunking Logic and NLP Processing Phase | Milestone 2 (End of Week 5):** Creation of the logic to divide the text into more manageable sections depending on predetermined parameters like word count or character count. Along with that Processing each document chunk by incorporating Hugging Face Transformers or other NLP libraries. Putting tokenization, preprocessing, and chunk-specific embeddings into practise. Setting up a mechanism to store the document chunks and associated embeddings for a short period.
* **Documentation, Review & Clean Up Phase | Milestone 3 (End of Week 6):** Designing and implementing the endpoints and developing the download functionality**.** Identifying and fixing any bugs or performance issues. Refining the implementation based on user feedback and identified improvements.

# Timeline

#### May 15th - June 11th (Before the Coding Period)

* Denotes the time between the announcement of selected contributors and before the commencement of the coding period on Jun 20
* Interact with mentors and polish timeline and milestones further.
* Get to know and understand project deliverables better, decide upon regular meetings, and have a great one-on-one interaction while asking for feedback.

#### June 20th - June 26th (Week 1)

* Requirement Analysis: Gathering of detailed requirements for the document uploader API.
* Write modules in python for easy interaction
* Designing the API endpoints ,request/response structures, and overall architecture of the project. Determine the necessary parameters for document upload, chunking, and embedding generation.

#### - End of Week 1 (Milestone 1 achieved)

**June 27th - July 10th (Week 2 and Week 3)**

* Implemention of the functionality to accept document uploads from users.
* Setting up the necessary infrastructure to handle file uploads or text input.
* Development of the logic to split the document into smaller chunks based on defined criteria such as word count or character count.

#### July 11th - July 17th (Week 4)

Integration of the Hugging Face Transformers or other NLP libraries to process each document chunk.

* Implemention of the tokenization, preprocessing steps, and generating embeddings for each chunk.
* Setting up a mechanism to store the document chunks and associated embeddings for a short period.
* Ensuring efficient memory usage and consider any security or privacy considerations by **the end of Week 4.**

#### July 18th - July 24th (Week 5)

* Quickly design a simple Figma template, and get approval from the mentor
* Designing and implementing an endpoint that allows users to retrieve the generated embeddings.
* Developing the feature that enables users to request downloads of the processed document or the embeddings.
* Implementing a mechanism to generate downloadable files or provide download links.
* **End of Week 5 (Milestone 2 achieved)**

**July 25th - July 31st (Week 6)**

* Implement documentation, Unit Testing, Integration Testing and perform final code clean-up.
* **End of Week 6 (Milestone 3 achieved)**

# Future Development

For handling huge documents and handling user requests, an asynchronous document embedding uploader API offers increased responsiveness, scalability, resource utilisation, fault tolerance, and flexibility. These benefits help create a system that is more effective and reliable, can manage a variety of workloads, and offers a smooth user experience.

* Advanced NLP Techniques: Explore and incorporate advanced NLP techniques into the project. This could involve leveraging transformer models for tasks like document summarization, sentiment analysis, entity recognition, or question answering. By expanding the range of NLP capabilities, you can provide users with more comprehensive and insightful analysis of their documents.
* Multi-Language Support: Extend the project to support multiple languages. This would involve training or integrating models specifically designed for different languages, enabling users to process and analyze documents in languages other than English. Supporting multilingual documents can significantly broaden the project's user base and make it more versatile.
* Custom Embedding Models: Allow users to train custom embedding models using their own labeled data. This would involve providing a mechanism to fine-tune pre-trained models or train entirely new models based on specific user requirements. Customization capabilities empower users to tailor the embeddings to their domain-specific needs and improve the accuracy of their document analysis.
* Integration with Search Engines and Knowledge Graphs: Integrate the project with search engines or knowledge graph systems to provide users with more comprehensive and contextualized search capabilities. By leveraging external knowledge bases, you can enhance the understanding and analysis of the documents, enabling users to explore related concepts, entities, and relationships.

This list is by no means comprehensive. By itself, the fields of AI and NLP are presently undergoing rapid and revolutionary advances. This project will continue to be developed in accordance with the principles of open source and will not only be done during the program's coding phase.

# Availability

#### Q. When do your classes and exams finish?

- I just finished my third year's examinations, therefore I won't be taking any more for the next four months. My classes will start in an online format on July 3rd. They can often go till 5 o'clock at the most, but that won't be a problem because I typically get up early and conduct the majority of my coding-related work in the morning. It wouldn't be difficult to manage the programme, the classes, and one-on-one mentor contacts.

#### Q. Do you have a full or part-time job or internship planned for this summer?

- No. I do not have any full or part-time job or internship planned for this entire summer and hence would be available for the project.

#### Q. How many hours per week do you have available for a summer project?

- For the summer project, I will be able to consistently put in an average of 60 hours per week. This initiative and the prospects it presents to me are both very important to me. I want to be a part of this fantastic initiative and see it through to the end.

# Personal Background

## About Me

Sanchari Ray

My name is Sanchari Ray and I am a 4th year BTech student. I've been coding for a while, and I've always been interested in computer science. I enjoy taking part in hackathons to put my newfound knowledge of frameworks and tech stacks to the test. I am familiar with a wide range of tech stacks, including those in the disciplines of data science, web development, machine learning, and cyber security. Working with startups and open source organisations is one of my favourite things to do since they push me to build things from the ground up and constantly teach me new things.

I adore Sir Ratan Tata as well, and I hope to meet him eventually. I enjoy being current with the newest technology, reading books, listening to podcasts, networking with others, and attempting to learn as much as I can. I believe artificial intelligence is the future, so I devote a lot of time to learning more about AI-tools, machine learning, and deep learning. I enjoy reading about the newest and most popular businesses around the world, as well as algorithms, astrophysics, and bit of blockchain technology.

I genuinely believe that it is our responsibility to improve the world, provide answers, and address issues in an effort to make human life more effective. I want to contribute in this growth and I'm hoping something wonderful will come out of it.

***LinkedIn:*** [*https://www.linkedin.com/in/sanchari-ray-a4499b21b/*](https://www.linkedin.com/in/sanchari-ray-a4499b21b/)

***Github:*** [*https://github.com/Sanchariii*](https://github.com/Sanchariii)

***Personal Website:*** [*https://sanchariii.github.io/portfolio-website/*](https://sanchariii.github.io/portfolio-website/)

## Education

I am currently pursuing my 4th year of BTech in Computer Science and Engineering at Kalinga Institute of Industrial Technology (KIIT), Bhubaneswar. I did my schooling at Aditya Birla Vani Bharati, Hooghly, West bengal(Class 1 - 12). I scored a total of 89.0 % in my class 10 exams, and a total of 89.1 % in my class 12 exams. Currently, I am focusing mainly on DataScience and ML-based technologies and am looking forward to pursuing my MBA degree in the future from abroad.

## Relevant work experience with the proposed tech stack

My entire life I have always wanted to be an entrepreneur. I feel learning to code well and achieving a certain level of mastery over the subject can make my journey significantly easier. I love volunteering my skills for society wherever and whenever possible.

I have worked on multiple projects based on DataScience, Machine Learning using AI-Tools,Web Development and made it to a success. Recently I was given a responsibility as a Project Manager in my College Tech Society(GDSC) and there I have made a more or less same project using Hugging Face Transformer and it was a success. This experience is what I believe would help me pull off this challenging project and thereby justifies my candidature.

## Any prior experience with open source development?

Yes. I have been into Open Source since my 2nd year. I have participated in Hacktoberfest and GSoC. Since I was a beginner I was learning and thus could not provide a very active participation, but I have now started actively contributing to multiple projects such as PyG project by IVY organization. I am looking forward to an exponential increase in my contributions to open source and hope that I learn a ton during the process.

.

## Why me?

I genuinely think that having familiarity with the relevant tech stack that is used in the proposed system will be extremely helpful to me in completing this difficult job. What motivates me and will continue to motivate me throughout the entire project is my enthusiasm and experience in building unique systems from scratch. I think I can learn a lot and give a lot to the community as a whole while working with my mentor and getting to know all the other mentors and beautiful people who were chosen for the program. I've worked on a variety of teams over the past two years, so I can quickly establish solid coordination with my teammates while also possessing conflict resolution and communication skills.

I feel that all of the above-mentioned reasons may be allowed to justify my acceptance as a contributor to the C4GT 2023 program under the Project Idea: **Document Uploader.**

#### Thanking Note

I want to express my sincere gratitude to all the mentors involved and the entire community for giving me the chance to work on such a creative project. I'll make an effort to give this community everything I've got. Thank you for putting up with all of my questions while I was writing this proposal, and I appreciate you all providing mentorship and direction as we develop our knowledge and talents. I'm eager to learn new things and look forward to the beautiful trip that lies ahead.