Epitome - Let's Make it Easy

A Major Project Report Submitted to

Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal
Towards Partial Fulfilment for the Award of

Bachelor of Technology
(Computer Science and Engineering)

Submitted By:
Sarthak Parakh (0827CS171196)
Shivam Goyan (0827CS171204)
Somya Jain (0827CS171215)
CS4 IV YEAR

Supervised & Guided by:
Prof. Kavita Namdev
Senior Assistant Professor

Department of Computer Science and Engineering
Acropolis Institute of Technology and Research, Indore
Jan - Jun 2021

Examiner Approval

The Project entitled ***"Epitome - Let's Make it Easy"*** submitted by **Sarthak Parakh (0827CS171196), Shivam Goyan (0827CS171204) and Somya Jain (0827CS171215)** has been examined and is hereby approved towards partial fulfillment for the award of ***Bachelor of Technology degree in Computer Science & Engineering*** discipline, for which it has been submitted. It understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it has been submitted.

(Internal Examiner)	**(External Examiner)**
Date:	**Date:**

Guide Recommendation

This is to certify that the work embodied in this project entitled "***Epitome - Let's Make it Easy***" submitted by **Sarthak Parakh (0827CS171196), Shivam Goyan (0827CS171204) and Somya Jain (0827CS171215)** is a satisfactory account of the bonafide work done under the supervision of ***Prof. Kavita Namdev***, is recommended towards partial fulfillment for the award of the Bachelor of Technology (Computer Science & Engineering) degree by Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal.

(Project Guide & Coordinator)

Student Undertaking

This is to certify that project entitled "***Epitome - Let's Make it Easy***" has developed by us under the supervision of ***Prof. Kavita Namdev.*** The whole responsibility of work done in this project is ours. The sole intension of this work is only for practical learning and research.

We further declare that to the best of our knowledge, this report does not contain any part of any work which has been submitted for the award of any degree either in this University or in any other University / Deemed University without proper citation and if the same work found then we are liable for explanation to this.

Sarthak Parakh (0827CS171196)

Shivam Goyan (0827CS171204)

Somya Jain (0827CS171215)

Acknowledgement

We thank the almighty Lord for giving us the strength and courage to sail out through the tough and reach on shore safely.

There are number of people without whom this projects work would not have been feasible. Their high academic standards and personal integrity provided us with continuous guidance and support.

We owe a debt of sincere gratitude, deep sense of reverence and respect to our guide and mentor **Prof. Kavita Namdev,** Sr. Asst. Professor, AITR, Indore for hermotivation, sagacious guidance, constant encouragement, vigilant supervision and valuable critical appreciation throughout this project work, which helped us to successfully complete the project on time.

We express profound gratitude and heartfelt thanks to **Dr. Kamal Kumar Sethi**, HOD CSE, AITR Indore for his support, suggestion and inspiration for carrying out this project. We are very much thankful to other faculty and staff members of CSE Dept, AITR Indore for providing us all support, help and advice during the project. We would be failing in our duty if do not acknowledge the support and guidance received from **Dr. S.C. Sharma**, Director,AITR, Indore whenever needed. We take opportunity to convey our regards to the management of Acropolis Institute, Indore for extending academic and administrative support and providing us all necessary facilities for project to achieve our objectives.

We are grateful to **our parents** and **family members** who have always loved and supported us unconditionally. To all of them, we want to say "Thank you", for being the best family that one could ever have and without whom none of this would have been possible.

Sarthak Parakh (0827CS171196), Shivam Goyan (0827CS171204), Somya Jain (0827CS171215)

Executive Summary

Epitome - Let's Make it Easy

This project is submitted to Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal(MP), India for partial fulfillment of Bachelor of Technology in Computer Science & Engineering branch under the sagacious guidance and vigilant supervision of ***Prof. Kavita Namdev***.

Epitome is an application which will take offline and online class/meeting audio as an input, convert it into a text document and will summarize the whole discussion. Text summarization is the process of generating short, fluent, and most importantly accurate summary of a respectively longer text document. The main idea behind automatic text summarization is to be able to find a short subset of the most essential information from the entire set and present it in a human-readable format. As online textual data grows, automatic text summarization methods have potential to be very helpful because more useful information can be read in a short time.

Key words : Text Summarization, summary

Table of Contents

Chapter 1. Introduction

A summary is a condensed version of the original text, which conveys vital information in short while preserving its key meaning. Since manual text summarization is a tedious task that can be biased, the automation of text summarization is gaining traction and tends to be a bold reason for academic research.

Automatic text summarization is a process of minimizing a band of data computationally, to generate a subset that carries the crucial and significant information from the original text with its essential meaning. Moreover, images and videos can also be summarized. As text summarization finds the most relevant sentences from the document, image summarization finds the most relevant image from the image pool, video summarization extracts the crucial frames from the video content. The most important advantage of using a text summarizer is that it increases readability and reduces the time investment. In general, automatic text summarizers select important sentences from the document and organize them together. The goal is to generate a shorter version with the same overall meaning of the document. Automatic text summarization is prevalent in the field of Natural Language Processing (NLP).

1.1 Overview

Epitome is an application which will take offline and online class/meeting audio as an input, convert it into a text document and will summarize the whole discussion. Text summarization is the process of generating short, fluent, and most importantly accurate summary of a respectively longer text document. The main idea behind automatic text summarization is to be able to find a short subset of the most essential information from the entire set and present it in a human-readable format. As online textual data grows, automatic text summarization methods have potential to be very helpful because more useful information can be read in a short time.

1.2 Background and Motivation

Today we know that machines have become smarter than us and can help us with every aspect of life. The technologies have reached to an extent where they can do all the tasks of human beings like household tasks, controlling home devices, making appointments etc. The field which makes these things happen is Machine Learning. Machine Learning trains the machines with some data which makes it capable of acting when tested by the similar type of data. The machines have become capable of understanding human languages using Natural Language Processing. Text analysis and summarization is a main part of Natural language processing.

1.3 Problem Statement and Objectives

Naturally, there is so much critical information being said via voice that gets lost once we leave the meeting room. No matter how hard we try to multitask, it's impossible to remember everything, type everything, and also stay present while the meeting is happening.

Thus, the system implemented has the following objectives:

1. Linking professionals and students to the new age technology.
2. To provide instant notes of offline and online meetings/classrooms.
3. Bringing more attention in meetings/classrooms.
4. We aim to provide a basic interface which can be used by anyone during online/offline meetings by providing them the summaries of the text from which they want to gain information.
5. Save the time.

1.4 Scope of the Project

The scopes of this project are:

- Performing Speech Recognition.
- Converting audio into text.
- Performing Natural language processing Algorithm for text skimming
- Providing text summarization.

1.5 Team Organization

- **Shivam Goyan :**

Along with doing preliminary investigation and understanding the limitations of the current system, I studied about the topic and its scope and surveyed various research papers related to the text summarization and the technology that is to be used.

I also worked on the implementation of the Summarization framework and the working of audio to text conversion in the project. Documentation is also a part of the work done by me in this project.

- **Sarthak Parakh :**

I investigated and found the right technology and studied it. For the implementation of the project, I collected the object data and created the model for it. Implementation logic for the project objective and coding of internal functionalities is also done by me.

I also did some documentation work along with surveying the research papers in order to increase the efficiency of the model created.

- **Somya Jain :**

I investigated and found the right technology and studied it. For the implementation of the project , I collected the object data and trained the model for it. Implementation logic for the project objective and coding of internal functionalities is also done by me.

I worked on creating a database for storing results in the database. Also, worked on Back end design for storing results in the database for maintaining logs.

1.6 Report Structure

The project ***Epitome- Let's make it easy*** is primarily concerned with the **text summarization in real-time** and the whole project report is categorized into five chapters.

Chapter 1: Introduction- introduces the background of the problem followed by rationale for the project undertaken. The chapter describes the objectives, scope and applications of the project. Further, the chapter gives the details of team members and their contribution in development of the project which is then subsequently ended with a report outline.

Chapter 2: Review of Literature- explores the work done in the area of Project undertaken and discusses the limitations of the existing system and highlights the issues and challenges of the project area. The chapter finally ends up with the requirement identification for present project work based on findings drawn from reviewed literature and end user interactions.

Chapter 3: Proposed System - starts with the project proposal based on requirement identified, followed by benefits of the project. The chapter also illustrates the software engineering paradigm used along with different design representations. The chapter also includes a block diagram and details of major modules of the project. Chapter also gives insights of different types of feasibility study carried out for the project undertaken. Later it gives details of the different deployment requirements for the developed project.

Chapter 4: Implementation - includes the details of different Technology/ Techniques/ Tools/ Programming Languages used in developing the Project. The chapter also includes the different user interfaces designed in the project along with their functionality. Further it discusses the experiment results along with testing of the project. The chapter ends with evaluation of the project on different parameters like accuracy and efficiency.

Chapter 5: Conclusion - Concludes with objective wise analysis of results and limitation of present work which is then followed by suggestions and recommendations for further improvement.

Chapter 2. Review of Literature

Text summarization is a process of retrieving crucial information in a concise and precise manner from original and voluminous texts while maintaining the overall meaning of the text. Data is growing exponentially day by day and so the textual data, which may be structured or unstructured, and the best way to use them is by skimming the results. We can access an immense amount of information, however, most of it is redundant, trivial, and may not deliver intended results. Using text summarization techniques can amplify the readability of text documents, reduce the investment of time in scrutinizing the information, and can increase the amount of information to be fitted in a particular domain.

2.1 Preliminary Investigation

2.1.1 Current System

Currently there is no existing similar application in the market, but there are various different types of text summarizers available using different-different approaches to Machine Learning. Some of the applications like fireflies.ai and chorus.ai transcribes and captures the text and summarizes it or have some features like recording web conferencing platforms and provides searches across the voice notes as well.

2.2 Limitations of Current System

On the economic front, it is not feasible for the person who is finding an easier and cheaper solution for their text and voice summarization having features to transcribe, record, search key facts and provide a summary out of it. The above platforms do provide adequate functionalities with a high price. They also collaborate with web conferencing platforms for high prices but Epitome acts like an Extension to these web conferencing platforms with very low rates and all the key functionalities to update, skim, transcribe, record and summarize the text or audio documents.

2.3 Requirement Identification and Analysis for Project

Significant work has been done in the field of Text Extraction and Summarization; however, it is not easy to achieve desired results. The review of literature leads to draw certain major findings which are as under :

Various technical documents were focused on single-document summarization. Luhn in 1958 shows the significance of words based on frequency measures. He deleted the stop words and rest words are given a hierarchy starting from root and index describing the significance of each word. This is calculated on the number of occurrences in a document called a significant factor and are ranked. Based on ranking, top sentences are selected to form a summary. Baxendale in 1958 focused on sentence position to find the salient features. He took 200 paragraphs and examined that in 85% of paragraphs topic sentences are placed in the beginning while in the rest 7% he found, it occurred in the last. Edmundson in 1969 proposed a typical structure that produces extracts. In the beginning he took around 400 technical documents and built a protocol producing manual extracts. He addressed the above two features (word frequency, word position) and gave the two new features named cue words and skeleton (title or heading). Also the weights were attached with these. He evaluated and found that 44% machine extracts matched with manual extracts. Various other pioneers were there who applied different techniques in single document summarization: In 1961 G.J. Rath used lexical indicators to determine the relevant information from documents. In 1995 Julian Kupiec used an algebraic method to determine different features like uppercase words, length, position of words by using naïve-bayes classifier. In 1997 ChinYew Lin determined the position of sentences by using algebraic methods. In 1999 Eduard Hovy used symbolic word knowledge with strong NLP processing to show the concepts relevancy. In 2005 S.P Yong used a neural network. He showed Summarization = Text pre-processing subsystem + Keywords Extraction sub-system + Summary production sub-system. In 1976 M.A. K. Halliday used lexical semantic relationships to build lexical cohesion blocks and their patterns. In 1984 Ruqaiya Hasan used lexical cohesion to identify similarity chains.

In 1988 William C.Mann used RST (rhetorical structure theory) to encode the terminal nodes of a tree. In 1991 Jane Morris used cohesion chains to determine the sequence of associated words. In 1997 Branimir Boguraev used saliency based content characterization to rank the important sentences in unstructured documents. In 2010 Li Chengcheng used RST to analyze candidate sentences, discover rhetoric relations and give the construction. In 2000 Hongyan Jing used the human abstraction concept by taking the closely related sentences and eliminating the extra ones. Multi-Document Summarization: The major contribution was done by McKeown and Raedev in 1995 (NLP group) at Columbia University and SUMMONS was built. Similarity measures were used and extractive techniques were applied. McKeown et al. in 1999 and Radev et al. in 2000 identified common themes using clustering while Barzilay et al. in 1999 produced composite sentences from clusters whereas Carbonell and Goldstein in 1998 used maximal marginal relevance (MMR). A major contribution where multi-document summarization was concatenated to a multilingual environment by Evans in 2005.

Various other pioneers have worked in this field using different techniques. G.Salton in 1989 used TFI X IDFI techniques to evaluate the frequency. Jun'ichi Fukumoto in 2004 generated an abstract by using TF/IDF for single and multiple documents. You Ouyang in 2009 used word hierarchical technique for most frequent terms at the top. Vikrant Gupta in 2012 used a kernel which serves as a guideline to choose other sentences for summary by using statistical measures. Inderjeet Mani in 1997 used a graph based method to discover the nodes by applying a spreading activation technique. Rada Mihalcea în 2004 used a graph based method by adding a vertex for every sentence by creating links for similar sentences. Xiaojun Wan in 2008 used a graph based method by introducing a two-link graph for both sentences and documents. Kathleen McKeown in 1995 used a time based technique which focuses on how the trends of events change with respect to time. Shanmugasundaram Hariharan in 2012 used sentence correlation method where sentences are extracted on the basis of vote casting, scores and positions to get extracts. Tiedan Zhu in 2012 emphasized on logical closeness rather than topical-closeness using sentence correlation method.

Multi-Document Text Summarization: Since 1990's, single document extraction has moved to multiple document extraction in the domain of news articles. Various news articles like Google News, Columbia Newsblaster and News In Essence were inspired from multi-document summarization. Though a single document puts contradictory results by overlapping the information because of multiple documents availability. So the major focus on summary is that summary should follow the completeness, correctness, erroneous property.

Jade Goldstein in 2000 used clustering, coverage, anti redundancy and summary cohesion for minimizing redundancy and maximizing both relevance and diversity, Judith D.Schlesinger in 2008 combining clustering, linguistics, statistics for summarization by using clustering based method. Nitin Agarwal in 2011 used a query-oriented approach with an unsupervised approach with the help of clustering based method.

2.3.1 Conclusion

This chapter reviews the literature surveys that have been done during the research work. The related work that has been proposed by many researchers has been discussed. The research papers related to automatic text summarization and techniques from 1961 to 2011 have been shown which discussed different techniques and algorithms to summarize text and audio documents.

Chapter 3. Proposed System

3.1 The Proposal

The proposal is to create a web application that can work independently or on different online meeting tools and generate the summary or gist of the whole discussion, meeting or class being held. With growing digital media and textual data we don’t have much time to collect important information so via Epitome we can convert speech based audio data into its summarization or normally can skim the textual data.

It can also perform live recording of the audio so that we can get the skimmed results of the discussions being held. Generally we can add recorded audio files and Epitome will generate its respective summary which will be 30% of the provided data.

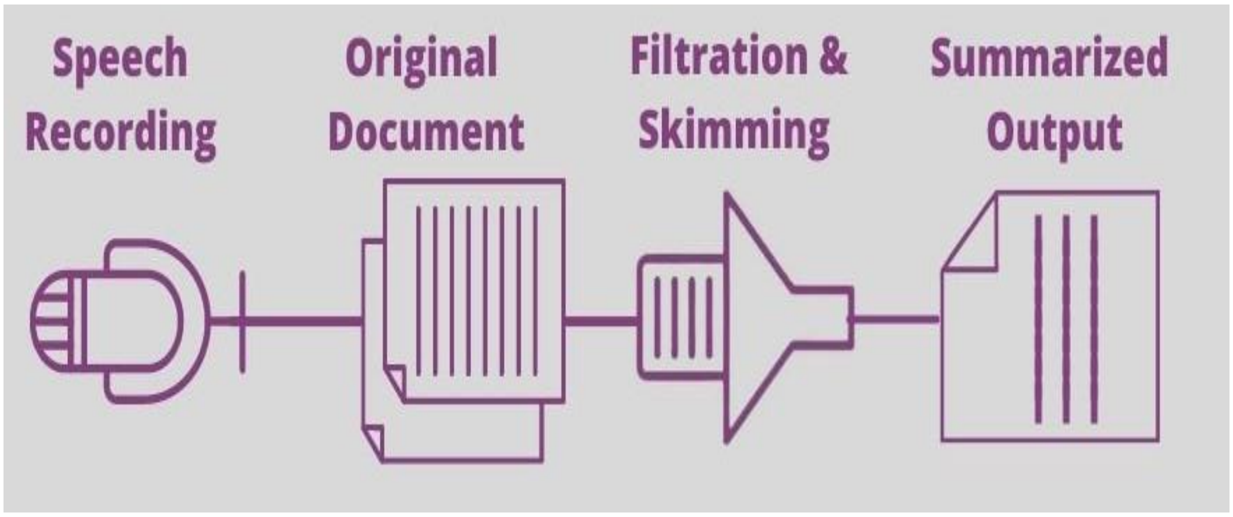
3.2 Benefits of the Proposed System

The current system had a lot of challenges that are overcome by this system:

1. Using text summarization techniques via Epitome can amplify the readability of text documents.
2. It will reduce the investment of time in scrutinizing the information.
3. It can increase the amount of information to be inserted in a particular domain.
4. Capture, clarify and consolidate key ideas quickly and easily
5. Retrieve your Notes wherever you go at the tap of a button.
6. No character limit
7. No Ads

- **Economic :** The proposed system is economic as there will not be any cost to perform summarization.
- **Real-Time Results:** Audio can be recorded anytime in the application as per requirement and can generate results, results are saved for later use as well.
- **Man Power :** It does not require any person or their efforts, just need to give input data.
- **24 x 7 Availability :** You can use the application anytime.
- **Statistical analysis :** The application is kept with a high end database to keep record of all the uploads add users and will be analyzed properly.

3.3 Block Diagram



Block Diagram

3.4 Feasibility Study

A feasibility study is an analysis of how successfully a system can be implemented, accounting for factors that affect it such as economic, technical and operational factors to determine its potential positive and negative outcomes before investing a considerable amount of time and money into it.

3.4.1 Technical

For an initial feasibility study, we investigated different extractive text summarization techniques such as TF-IDF, Cluster Based Method, Machine Learning Approach, Graph-Theoretic Approach, etc. whether machine learning techniques that had proved to be successful for the summarization of broadcast news, could be adapted for the meetings domain. A fully functional automatic meeting recording summarization system would be highly complex, since it combines high quality speech recognition, speaker segmentation, utterance segmentation, dialogue act interpretation, domain knowledge with summarization techniques, each of which components are not sufficiently mature yet. Therefore, we performed a limited study into the effectiveness of structural and lexical properties of utterances as features based on manual meeting transcripts.

3.4.2 Economical

For any real-time audio to text summarization system, there is a need for a High definition Microphone for better and accurate results.

Since the system is completely automated, there is a need for continuous electricity supply for it to operate 24X7.

The NLP framework used in the system works great with in any available device but a greater GPU will yield faster results.

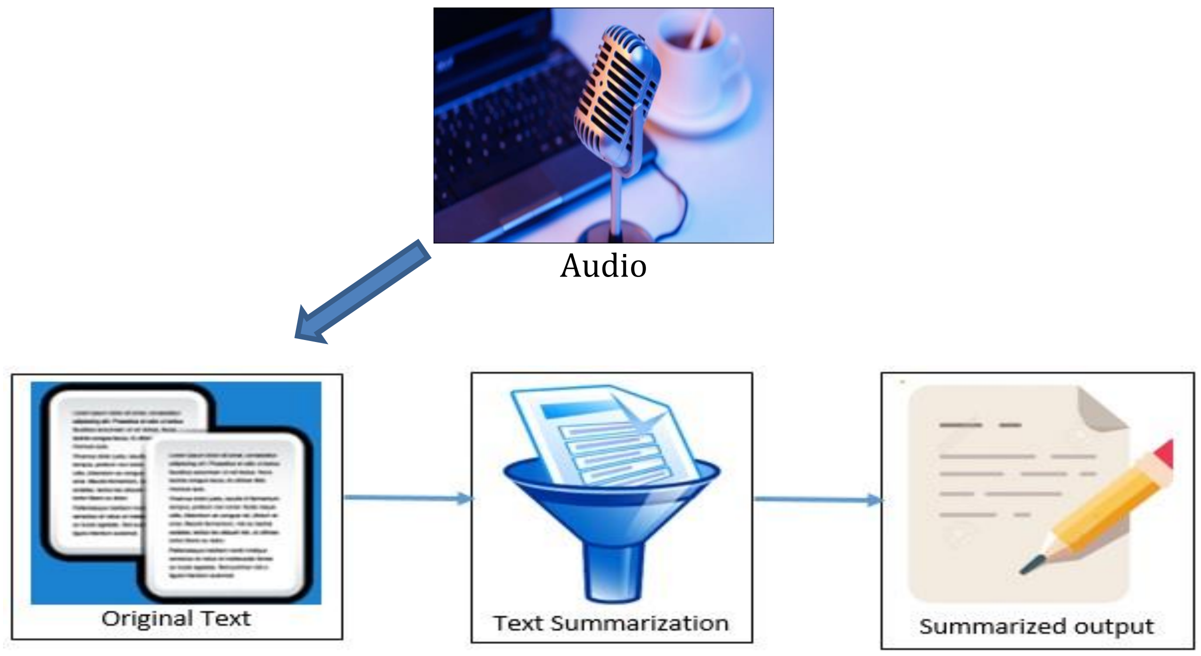
Since the system uses high performance processors continuously, to save any disaster from occurring due to very high temperatures, there is a requirement of a cooling system in the environment where it is implemented.

3.4.3 Operational

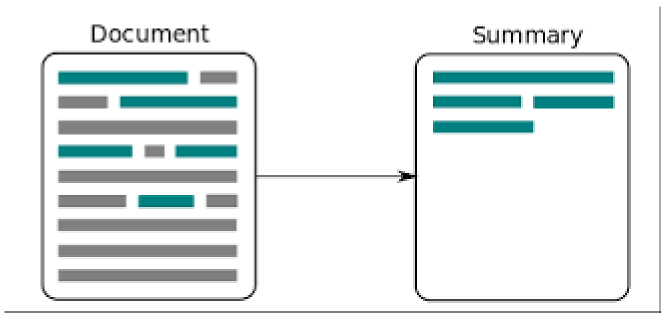
The main motto of our system is to reduce the manual efforts of searching, reading and collecting different textual resources, by generating shorter versions of these recordings or textual data. The application is able to do that accurately and efficiently making the application operationally feasible.

3.5 Design Representation

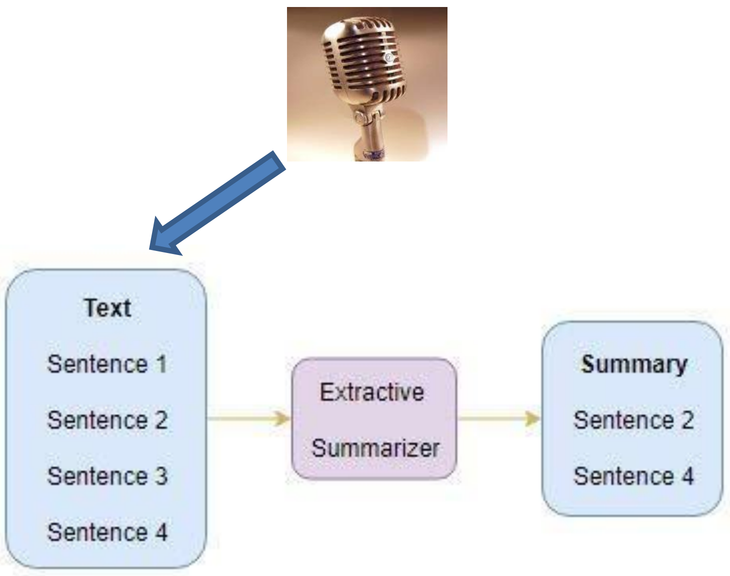
The design function of the application is diversified as it uses extractive summarization technique TF-IDF to perform action and summarize results.



Process Diagram

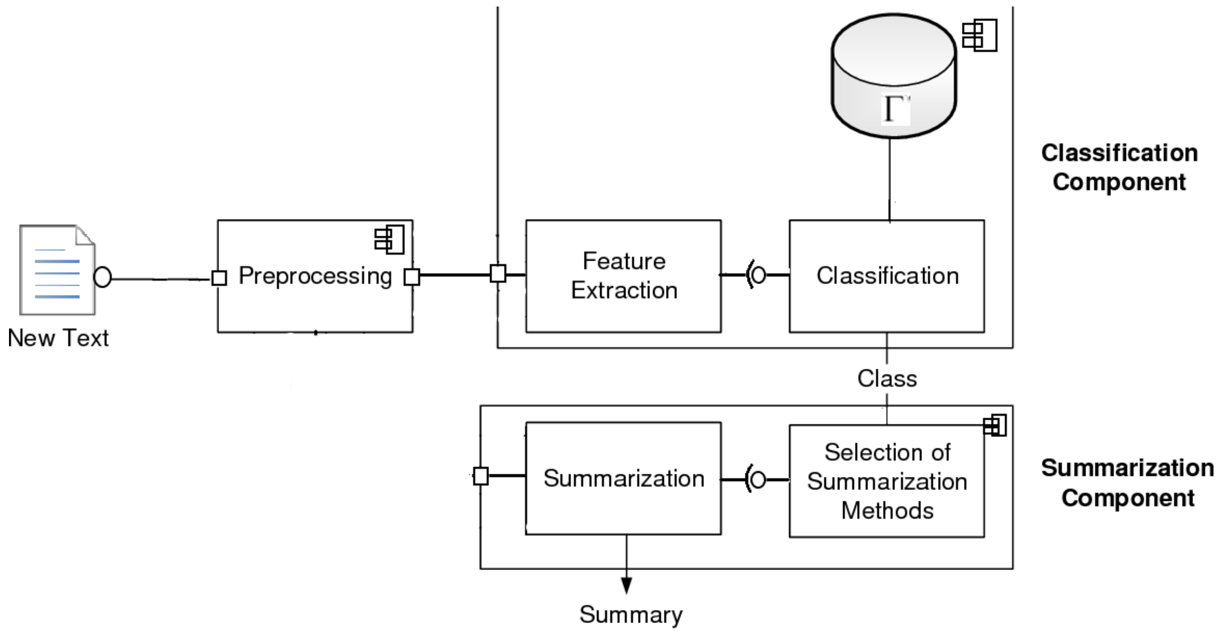


Summarization Diagram

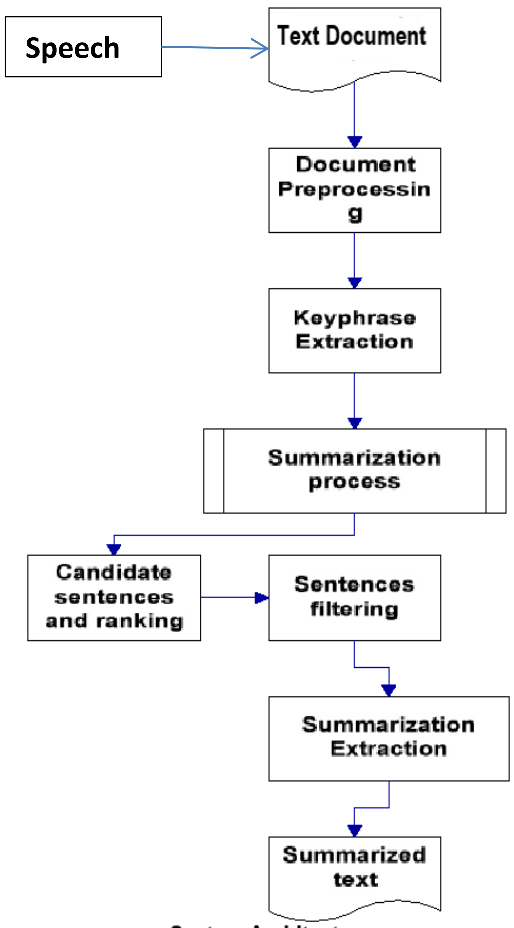


Extractive Technique

3.5.1 Component Diagram



3.5.2 System Architecture Diagram



System Architecture

3.5.3 Database Structure

DB Name - BASE_DIR / 'db.sqlite3'

Users Table
Username - CharField
Email - EmailField
First Name - CharField
Last Name - CharField
Staff Status - Boolean
Password - CharField

username	**(Required) 30 characters or fewer and can contain alphanumeric, _, @, +, . and - characters.**
first_name	**(Optional) 30 characters or fewer.**
last_name	**(Optional) 30 characters or fewer.**
email	**(Optional) Email address.**
password	**(Required) A hash of, and metadata about, the password. Note that Django doesn't store the raw password.**
groups	**A many-to-many relationship to django.contrib.auth.models.Group**
user_permissions	**A many-to-many relationship to django.contrib.auth.Permission**
is_staff (Boolean)	**Designates whether a user can access the admin site.**
is_active (Boolean)	**Designates whether a user is considered active.**

is_superuser	**(Boolean) Designates whether a user has all permissions without explicitly assigning them.**
last_login	**A datetime of the user's last login, set to NULL if the user has never logged in**
date_joined	**A datetime designating when the account was created. Is set to the current date/time by default when the account is created.**

Posts Table -
ID - BigAutoField
Title - CharField
Record - FileField
Content - TextField
Summary - TextField
Date Posted - DateTimeField
Author – User(foreign key)

Profile Table
ID - BigAutoField
User – User(foreign key)
Image - ImageField

3.6 Deployment Requirements

There are various requirements (hardware, software and services) to successfully deploy the system. These are mentioned below :

3.6.1 Hardware

- 32-bit, x86 Processing system
- Windows 7 or later operating system
- High processing computer system without GPU or with GPU(high performance)
- High- definition Audio Recorder/microphone

3.6.2 Software

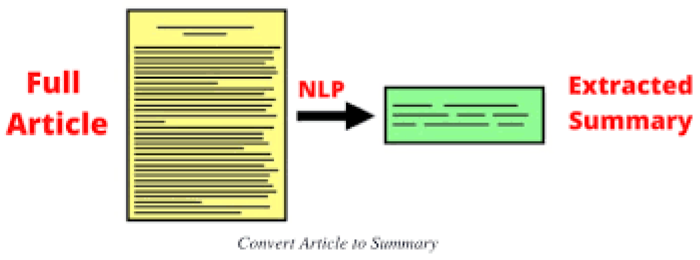
- NLP
- Python and its supported libraries

Chapter 4. Implementation

4.1 Technique Used

4.1.1 Extractive Summarization using TF-IDF

The name gives away what this approach does. We identify the important sentences or phrases from the original text and extract only those from the text. Those extracted sentences would be our summary. The below diagram illustrates extractive summarization:



Extractive Summarization using TF-IDF

A High weight in TF-IDF is reached by a high term frequency(in the given document) and a low document frequency of the term in the whole collection of documents.

TF-IDF algorithm is made of 2 algorithms multiplied together.

Term Frequency

Term frequency (TF) is how often a word appears in a document, divided by how many words there are.

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)

Inverse document frequency

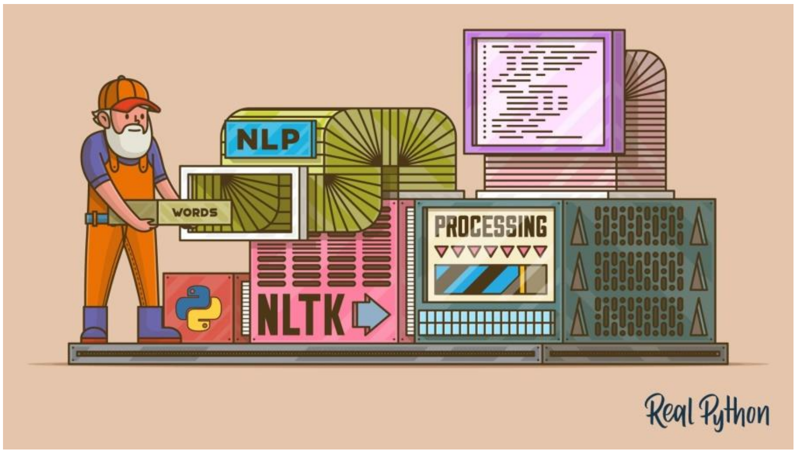
Term frequency is how common a word is, inverse document frequency (IDF) is how unique or rare a word is.

IDF(t) = log_e(Total number of documents / Number of documents with term t in it)

4.1.2 NLTK

The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).

It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports.



NLTK

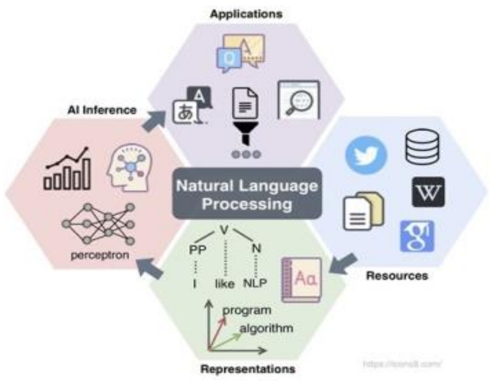
The Natural Language Toolkit is an open source library for the Python programming language originally written by Steven Bird, Edward Loper and Ewan Klein for use in development and education. It comes with a hands-on guide that introduces topics in computational linguistics as well as programming fundamentals for Python which makes it suitable for linguists who have no deep knowledge in programming, engineers and researchers that need to delve into computational linguistics, students and educators.

4.2 Tools Used

4.2.1 Natural Language Processing

Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data.

NLP makes it possible for computers to read text, hear speech, interpret it, measure sentiment and determine which parts are important.



Natural Language Processing

NLP draws from many disciplines, including computer science and computational linguistics, in its pursuit to fill the gap between human communication and computer understanding. Natural language processing helps computers communicate with humans in their own language and scales other language-related tasks. For example, NLP makes it possible for computers to read text, hear speech, interpret it, measure sentiment and determine which parts are important.

Today's machines can analyze more language-based data than humans, without fatigue and in a consistent, unbiased way. Considering the staggering amount of unstructured data that's generated every day, from medical records to social media, automation will be critical to fully analyse text and speech data efficiently.

4.2.2 Sound Device Python:

It is a Python tool to record and listen audio. It provides binding for the PortAudio library and a few convenience functions to play and record Numpy arrays containing audio signals.

4.3 Language Used

Python language is used in the system due to the following characteristics :

Simple :

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English (but very strict English!). This pseudocode nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the syntax i.e. the language itself.

Free and Open Source :

Python is an example of a FLOSS (Free/Libre and Open Source Software). In simple terms, you can freely distribute copies of this software, read the software's source code, make changes to it, use pieces of it in new free programs, and know that you know you can do these things. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and improved by a community who just want to see a better Python.

Object Oriented :

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simple way of doing object-oriented programming, especially, when compared to languages like C++ or Java.

Extensive Libraries :

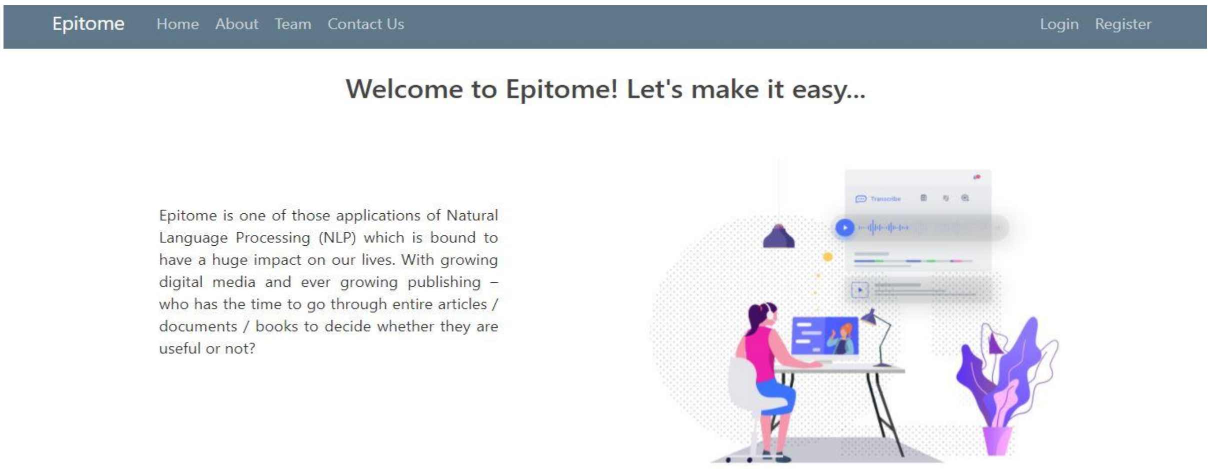
The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, ftp, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI(graphical user interfaces) using Tk, and also other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the "batteries included" philosophy of Python.

Frameworks :

Python provide different frameworks which are generally used. The framework used is our application is Django. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Django is a free and open-source web framework that follows the model–template–views (MTV) architectural pattern.

4.4 Screenshots

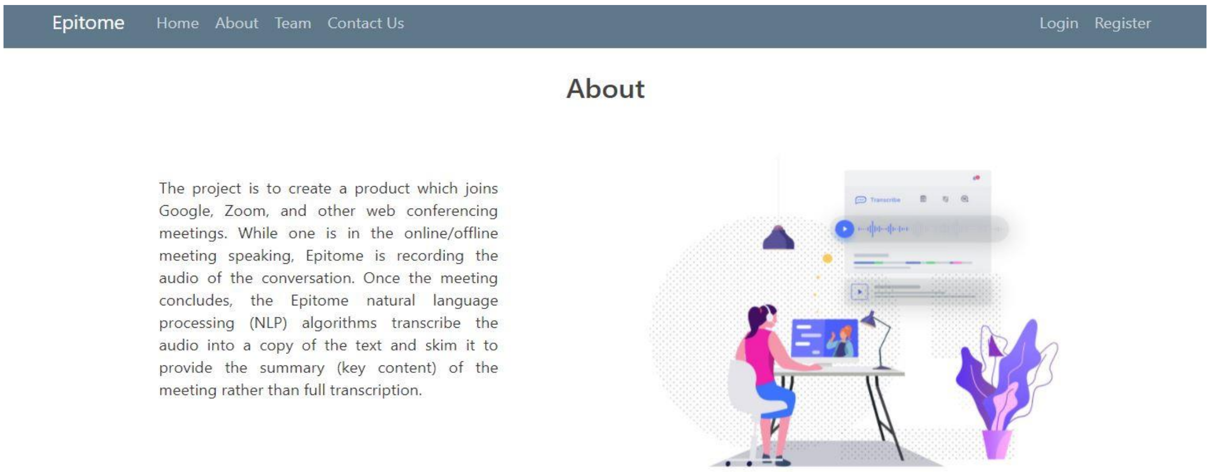
The Following are the screenshots of the result of the project :



Welcome to Epitome! Let's make it easy...

Epitome is one of those applications of Natural Language Processing (NLP) which is bound to have a huge impact on our lives. With growing digital media and ever growing publishing – who has the time to go through entire articles / documents / books to decide whether they are useful or not?

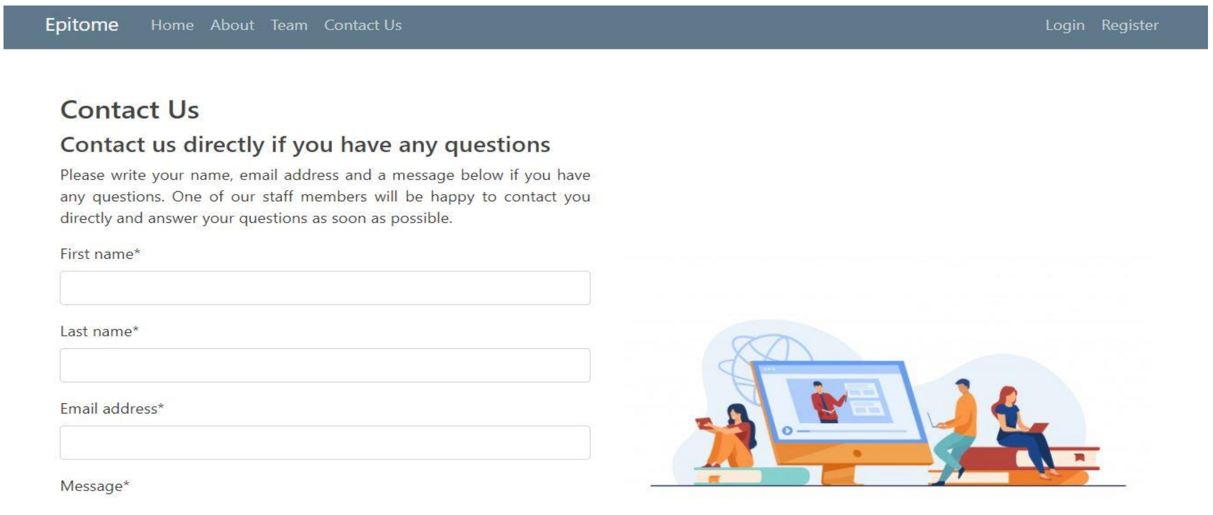
Screenshot 1



About

The project is to create a product which joins Google, Zoom, and other web conferencing meetings. While one is in the online/offline meeting speaking, Epitome is recording the audio of the conversation. Once the meeting concludes, the Epitome natural language processing (NLP) algorithms transcribe the audio into a copy of the text and skim it to provide the summary (key content) of the meeting rather than full transcription.

Screenshot 2



Contact Us

Contact us directly if you have any questions

Please write your name, email address and a message below if you have any questions. One of our staff members will be happy to contact you directly and answer your questions as soon as possible.

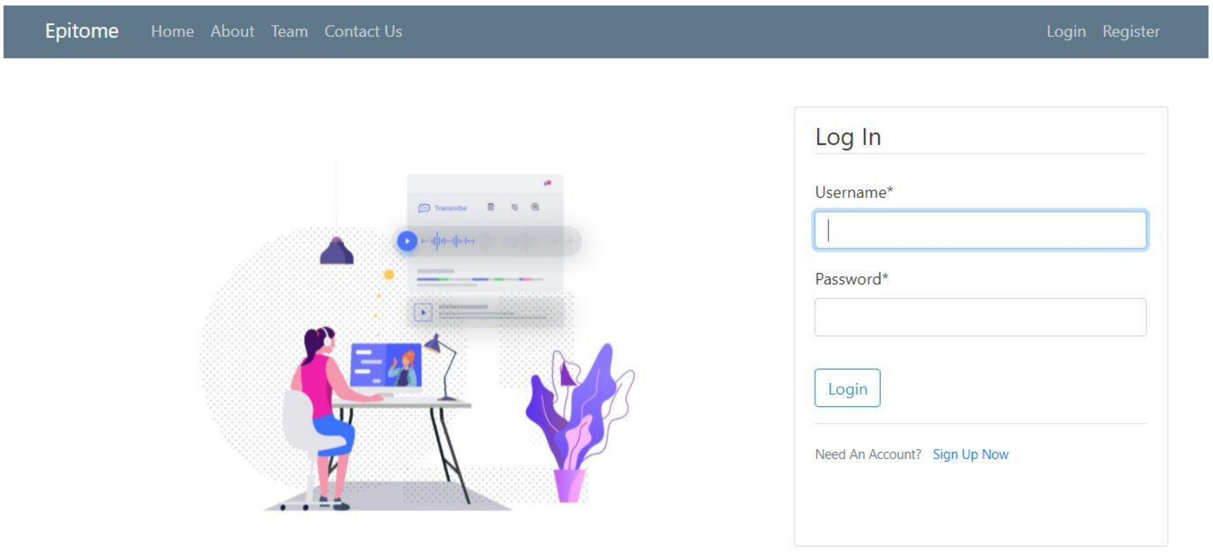
First name*

Last name*

Email address*

Message*

Screenshot 3



Log In

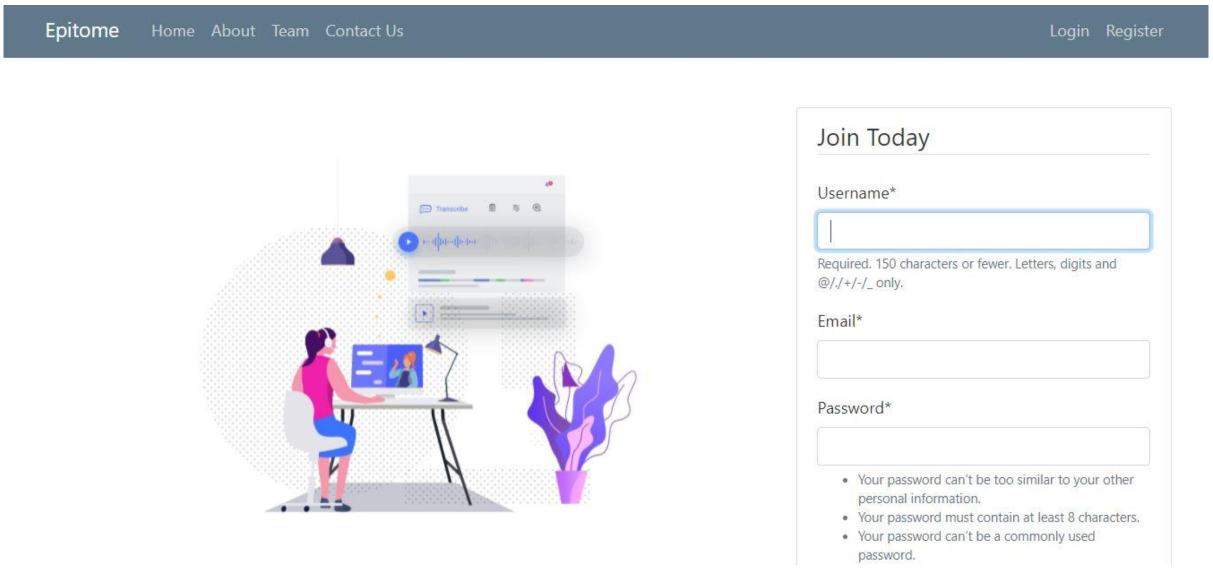
Username*

Password*

Login

Need An Account? Sign Up Now

Screenshot 4



Join Today

Username*

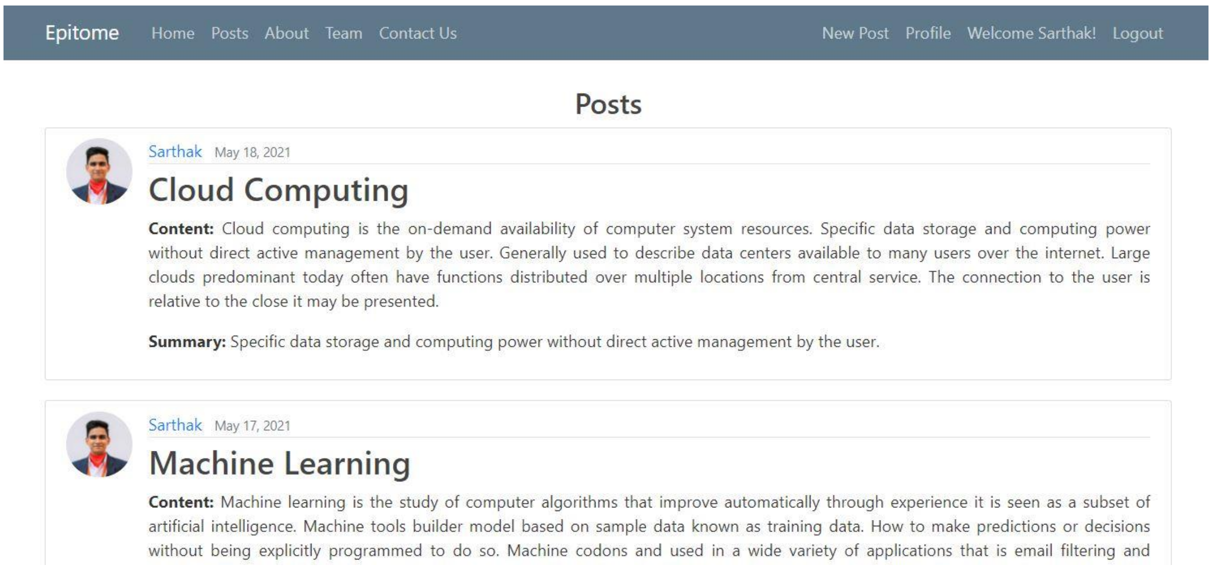
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.

Screenshot 5



Posts

Sarthak May 18, 2021

Cloud Computing

Content: Cloud computing is the on-demand availability of computer system resources. Specific data storage and computing power without direct active management by the user. Generally used to describe data centers available to many users over the internet. Large clouds predominant today often have functions distributed over multiple locations from central service. The connection to the user is relative to the close it may be presented.

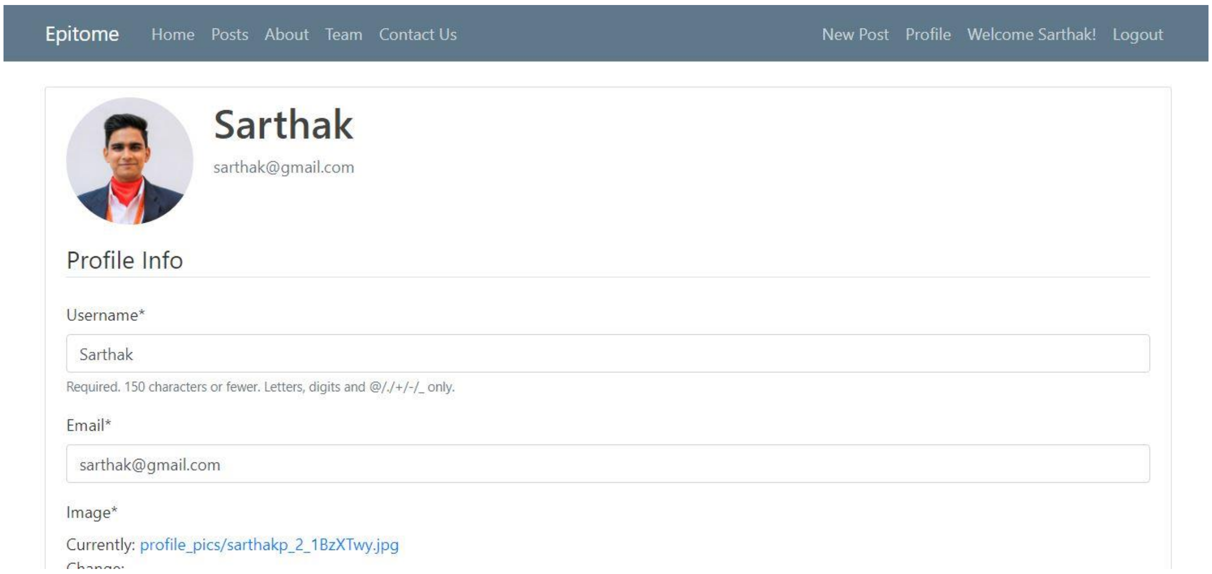
Summary: Specific data storage and computing power without direct active management by the user.

Sarthak May 17, 2021

Machine Learning

Content: Machine learning is the study of computer algorithms that improve automatically through experience it is seen as a subset of artificial intelligence. Machine tools builder model based on sample data known as training data. How to make predictions or decisions without being explicitly programmed to do so. Machine codons and used in a wide variety of applications that is email filtering and

Screenshot 6



Sarthak

sarthak@gmail.com

Profile Info

Username*

Sarthak

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

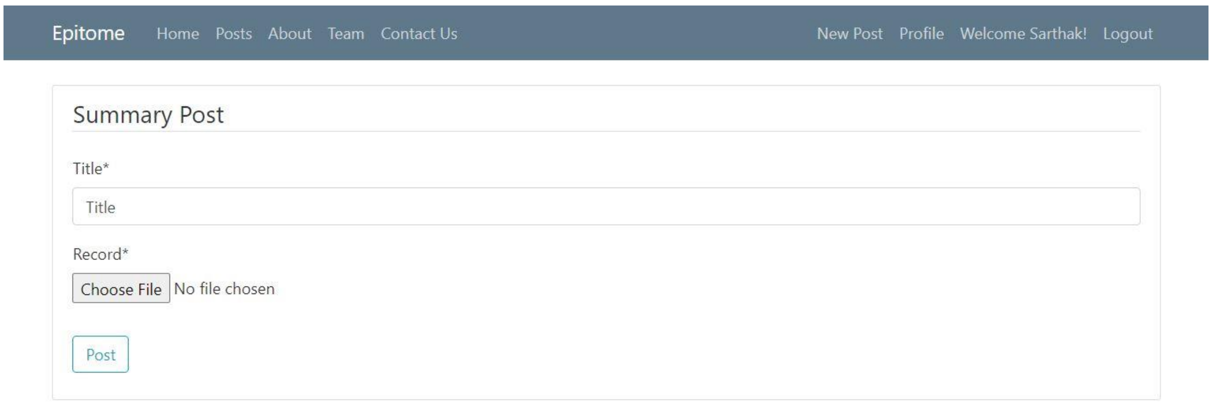
Email*

sarthak@gmail.com

Image*

Currently: profile_pics/sarthakp_2_1BzXTwy.jpg

Screenshot 7



Summary Post

Title*

Title

Record*

Choose File No file chosen

Post

Screenshot 8

4.5 Testing

Testing is the process of evaluation of a system to detect differences between given input and expected output and also to assess the features of the system. Testing assesses the quality of the product. It is a process that is done during the development process. .

4.5.1 Strategy Used

Tests can be conducted based on two approaches –

- Functionality testing
- Implementation testing

The texting method used here is Black Box Testing. It is carried out to test functionality of the program. It is also called 'Behavioral' testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested 'ok', and problematic otherwise.

4.5.2 Test Case and Analysis

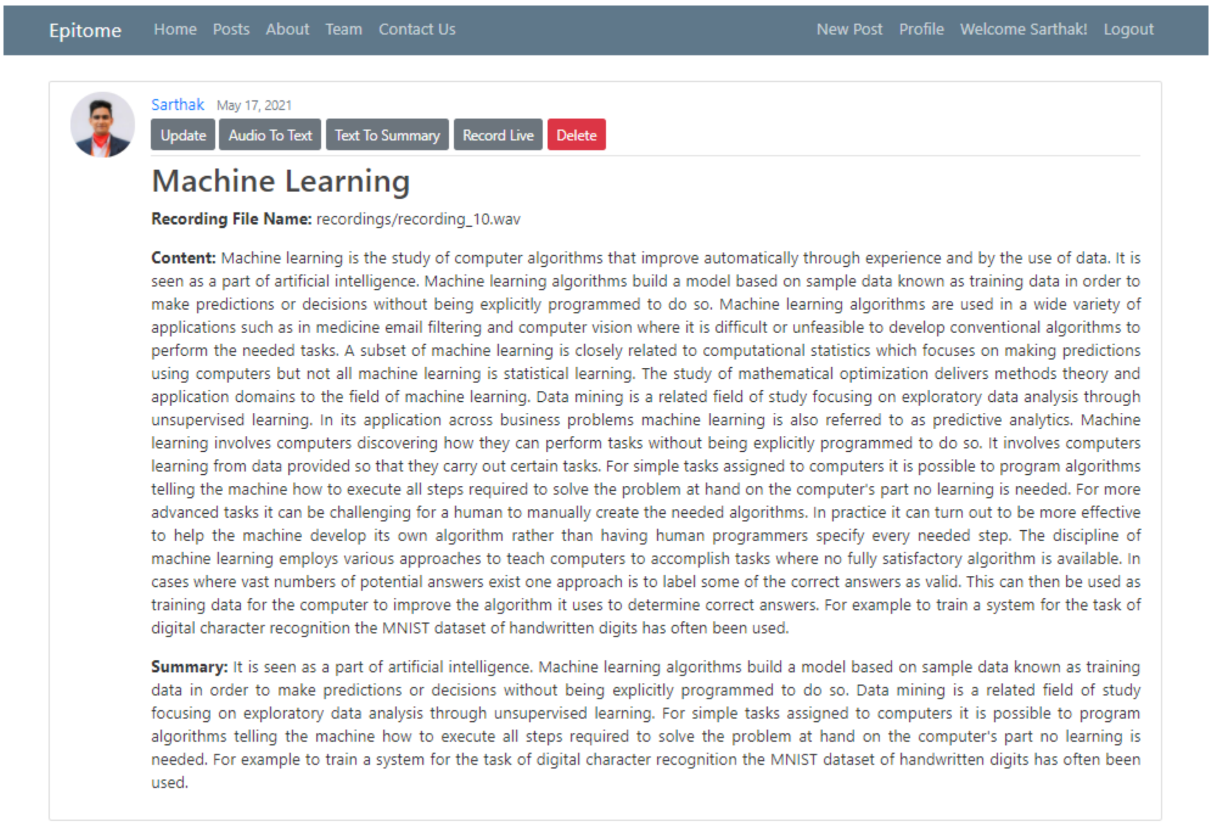
<u>TEST CASE: 1</u>

Input: recording_10.wav
Speech to text:
Content: Machine learning is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data known as training data in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications such as in medicine email filtering and computer vision where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks. A subset of machine learning is closely related to computational statistics which focuses on making predictions using computers but not all machine learning is statistical learning. The study of mathematical optimization delivers methods theory and application domains to

the field of machine learning. Data mining is a related field of study focusing on exploratory data analysis through unsupervised learning. In its application across business problems machine learning is also referred to as predictive analytics. Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand on the computer's part no learning is needed. For more advanced tasks it can be challenging for a human to manually create the needed algorithms. In practice it can turn out to be more effective to help the machine develop its own algorithm rather than having human programmers specify every needed step. The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithm it uses to determine correct answers. For example to train a system for the task of digital character recognition the MNIST dataset of handwritten digits has often been used.

TEST CASE 1 OUTPUT

Text to Summary: It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data known as training data in order to make predictions or decisions without being explicitly programmed to do so. Data mining is a related field of study focusing on exploratory data analysis through unsupervised learning. For simple tasks assigned to computers it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand on the computer's part no learning is needed. For example to train a system for the task of digital character recognition the MNIST dataset of handwritten digits has often been used.



Machine Learning

Recording File Name: recordings/recording_10.wav

Content: Machine learning is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data known as training data in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications such as in medicine email filtering and computer vision where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks. A subset of machine learning is closely related to computational statistics which focuses on making predictions using computers but not all machine learning is statistical learning. The study of mathematical optimization delivers methods theory and application domains to the field of machine learning. Data mining is a related field of study focusing on exploratory data analysis through unsupervised learning. In its application across business problems machine learning is also referred to as predictive analytics. Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand on the computer's part no learning is needed. For more advanced tasks it can be challenging for a human to manually create the needed algorithms. In practice it can turn out to be more effective to help the machine develop its own algorithm rather than having human programmers specify every needed step. The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithm it uses to determine correct answers. For example to train a system for the task of digital character recognition the MNIST dataset of handwritten digits has often been used.

Summary: It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data known as training data in order to make predictions or decisions without being explicitly programmed to do so. Data mining is a related field of study focusing on exploratory data analysis through unsupervised learning. For simple tasks assigned to computers it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand on the computer's part no learning is needed. For example to train a system for the task of digital character recognition the MNIST dataset of handwritten digits has often been used.

Test Case 1 output

<u>TEST CASE: 2</u>

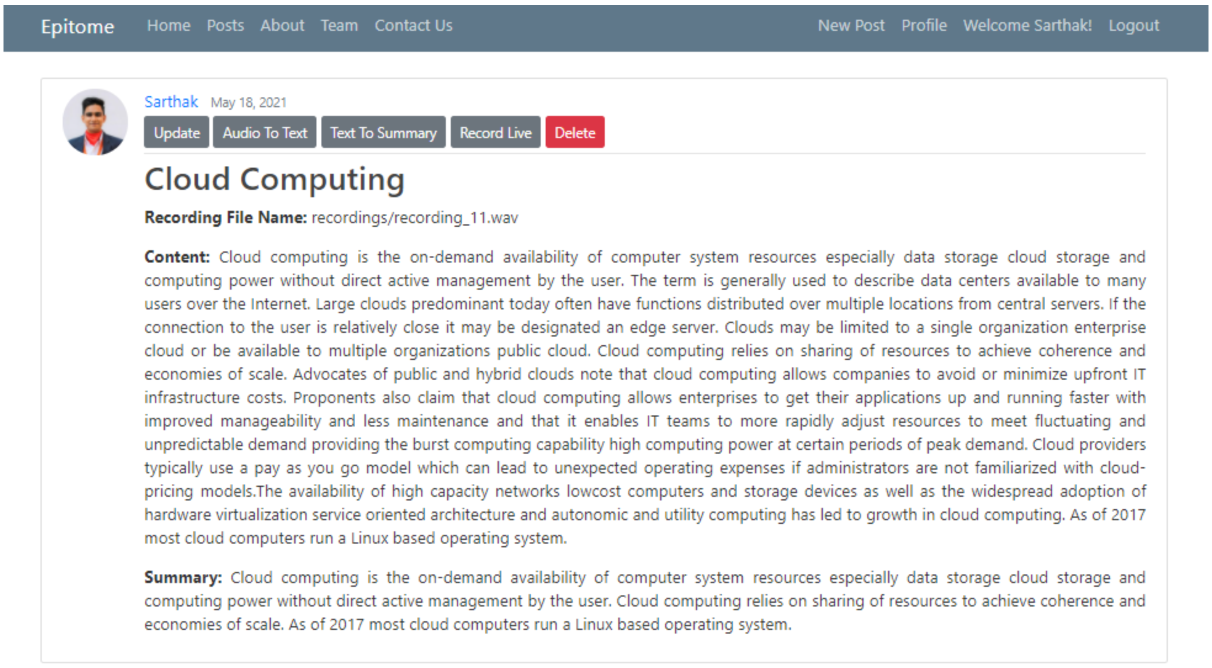
Input: recording_11.wav

Speech to text:

Content: Cloud computing is the on-demand availability of computer system resources especially data storage cloud storage and computing power without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet. Large clouds predominant today often have functions distributed over multiple locations from central servers. If the connection to the user is relatively close it may be designated an edge server. Clouds may be limited to a single organization enterprise cloud or be available to multiple organizations public cloud. Cloud computing relies on sharing of resources to achieve coherence and economies of scale. Advocates of public and hybrid clouds note that cloud computing allows companies to avoid or minimize upfront IT infrastructure costs. Proponents also claim that cloud computing allows enterprises to get their applications up and running faster with improved manageability and less maintenance and that it enables IT teams to more rapidly adjust resources to meet fluctuating and unpredictable demand providing the burst computing capability high computing power at certain periods of peak demand. Cloud providers typically use a pay as you go model which can lead to unexpected operating expenses if administrators are not familiarized with cloud-pricing models. The availability of high capacity networks low-cost computers and storage devices as well as the widespread adoption of hardware virtualization service oriented architecture and autonomic and utility computing has led to growth in cloud computing. As of 2017 most cloud computers run a Linux based operating system.

TEST CASE 2 OUTPUT

Text to Summary: Cloud computing is the on-demand availability of computer system resources especially data storage cloud storage and computing power without direct active management by the user. Cloud computing relies on sharing of resources to achieve coherence and economies of scale. As of 2017 most cloud computers run a Linux based operating system.



Cloud Computing

Recording File Name: recordings/recording_11.wav

Content: Cloud computing is the on-demand availability of computer system resources especially data storage cloud storage and computing power without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet. Large clouds predominant today often have functions distributed over multiple locations from central servers. If the connection to the user is relatively close it may be designated an edge server. Clouds may be limited to a single organization enterprise cloud or be available to multiple organizations public cloud. Cloud computing relies on sharing of resources to achieve coherence and economies of scale. Advocates of public and hybrid clouds note that cloud computing allows companies to avoid or minimize upfront IT infrastructure costs. Proponents also claim that cloud computing allows enterprises to get their applications up and running faster with improved manageability and less maintenance and that it enables IT teams to more rapidly adjust resources to meet fluctuating and unpredictable demand providing the burst computing capability high computing power at certain periods of peak demand. Cloud providers typically use a pay as you go model which can lead to unexpected operating expenses if administrators are not familiarized with cloud-pricing models.The availability of high capacity networks lowcost computers and storage devices as well as the widespread adoption of hardware virtualization service oriented architecture and autonomic and utility computing has led to growth in cloud computing. As of 2017 most cloud computers run a Linux based operating system.

Summary: Cloud computing is the on-demand availability of computer system resources especially data storage cloud storage and computing power without direct active management by the user. Cloud computing relies on sharing of resources to achieve coherence and economies of scale. As of 2017 most cloud computers run a Linux based operating system.

Test Case 2 Output

Chapter 5. Conclusion

5.1 Conclusion

The facts mentioned in the project reveal how Epitome can be used to fill major loopholes in online and offline meetings conducted globally. Despite the drawbacks, this system can be a major breakthrough in the digital, education and professional sector. Even in the worst conditions, with a few minor modifications, it will enable the optimization to reduce the obstacles that come their way. In short, Epitome is a step forward towards technology, innovation and hence overall growth.

5.2 Limitations of the Work

1. Language Barrier- We have made our speech recognizer for English only.
2. Usage of only a single language in speech i.e. only English.

5.3 Suggestion and Recommendations for Future Work

- Text Summarization through automation approaches depends upon the semantic analysis of the data in the document. So, in this way, the document sentences are gathered, and parameters are checked for word sense, keywords, similarities to the titles, words, and sentence occurrences, to sense utilities of covering sentences.

- The fusion of various techniques will provide the high potential results as a summarized output. With the ever-growing data in the environment, reduction of redundant and un-relevant data is very crucial, otherwise handling these data will become very difficult and will lead to loss of important information.

- Among future work is the use of all the more balanced gathering of the techniques to upgrade text summarization methodologies. Future researches will be made on diverse things like, morphological parsers, printed entailment and anaphoric assurance. Handwritten Text Programmatic content summarization for various text archives will record and reinvent short sentences for long archive files with disambiguation techniques will overcome the huge documents in the near future.

Bibliography

[1] D. K. Gaikwad, C. N. Mahender, "A Review Paper on Text Summarization", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 05, Issue 03, Mar 2016.

[2] P. Devihosur, Naseer R, "Automatic Text Summarization Using Natural Language Processing", International Research Journal of Engineering and Technology, Vol. 04, Issue 08, Aug 2017.

[3] Saranyamol CS, Sindhu L, "A Survey on Automatic Text Summarization", International Journal of Computer Science and Information Technologies, Vol. 5 (6), 2014.

[4] N. bhatia, A. Jaiswal, "Literature Review on Automatic Text Summarization: Single and Multiple Summarizations", International Journal of Computer Applications (0975 – 8887) Volume 117 – No. 6, May 2015.

[5] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E.D. Trippe, J.B. Gutierrez, K. Kochut, "Text Summarization Techniques: A Brief Survey", University of Georgia, Athens, GA, July 2017 Retrieved from: arXiv:1707.02268v3 [cs.CL].

[6] E. Hovy, C.Y. Lin, "Automated Text Summarization in Summarist", Information Sciences Institute of the University of California, Retrieved from: https://www.aclweb.org/anthology/W97-0704.pdf.

[7] W. Xiao, G. Carenini, "Extractive Summarization of Long Documents by Combining Global and Local Context", Computer Science University of British, Columbia, April 2019 Retrieved from: https://www.aclweb.org/anthology/D19-1298.pdf

[8] V. Qazvinian, D.R. Radev, S.M. Mohammad, B. Dorr, D. Zajic, M. Whidby, T. Moon, “Generating Extractive Summaries of Scientific Paradigms”, Journal of Artificial Intelligence Research 46 (2013) 165-201.

[9] G. Murray, S. Renals, J. Carletta, “Extractive Summarization of Meeting Recordings”, Centre for Speech Technology Research University of Edinburgh, 2004.

[10] D.D. Anbui, G.D. Fiol, J.F. Hurdle, S. Jonnalagadda, “Extractive Text Summarization system to aid data extraction from full text in systematic review development”, Journal of Biomedical Informatics, December 2016.

[11] K. D. Patil, S. A. Patil, Y. S. Deshmukh, “An Extractive Approach for English Text Summarization”, International Journal of Sciences & Applied Research, IJSAR, 6(5), 2019

[12] R. Barzilay, K. R. McKeown, M. Elhadad, “Information Fusion in the context of Multi-document Summarization”, Association for Computational Linguistics, Jun 1999, retrieved from: https://www.aclweb.org/anthology/P99-1071.pdf

[13] A. A. Natesh, S. T. Balekuttira, A. P. Patil, “Graph Based Approach for Automatic Text Summarization” International Journal of Advanced Research in Computer & Communication Engineering, Vol. 05, Special Issue 02, Oct 2016.

[14] J. L. Neto, A. A. Freitas, C. A. A. Kaedtner, “Automatic Text Summarization Using Machine Learning Approach”, 2002. SBIA '02: Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence.

[15] M. G. Ozsoy, F. N. Alpaslan, I. Cicekli , “Text Summarization using Latent Semantic Analysis”, Journal of Information Science, Jun 2011.

[16] Sarda A.T, Kulkarni A.R, “Text Summarization using Neural Networks and Rhetorical Structure Theory”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 04, Issue 06, Jun 2015.

[17] R. S. Dixit, Prof. Dr. S.S. Apte, "Improvement of Text Summarization using Fuzzy Logic Based Method" IOSR Journal of Computer Engineering, Vol. 05, Issue 06 Sep-Oct 2012.

[18] P.D. Patil, Prof. N.J. Kulkarni,"Text Summarization using Fuzzy Logic", International Journal of Innovative Research in Advanced Engineering (IJIRAE), Vol. 01, Issue 03, May 2014 (Special Issue).

[19] N. Vanetic, M. Litvak, "Query Based Summarization using MDL Principle", Association for Computational Linguistics ISBN 978-1-945626-41-8, retrieved from: https://www.aclweb.org/anthology/W17-1004.pdf

[20] A. Ajay, S. Rajendran, V. Shravan, "Query Based Text Summarization using Averaged Query", International Journal of Psychosocial Rehabilitation, June 24, 2020.

[21] A. Jain, "Automatic Extractive Text Summarization using TF-IDF", Web Blog Post - medium.com, Apr 2019.

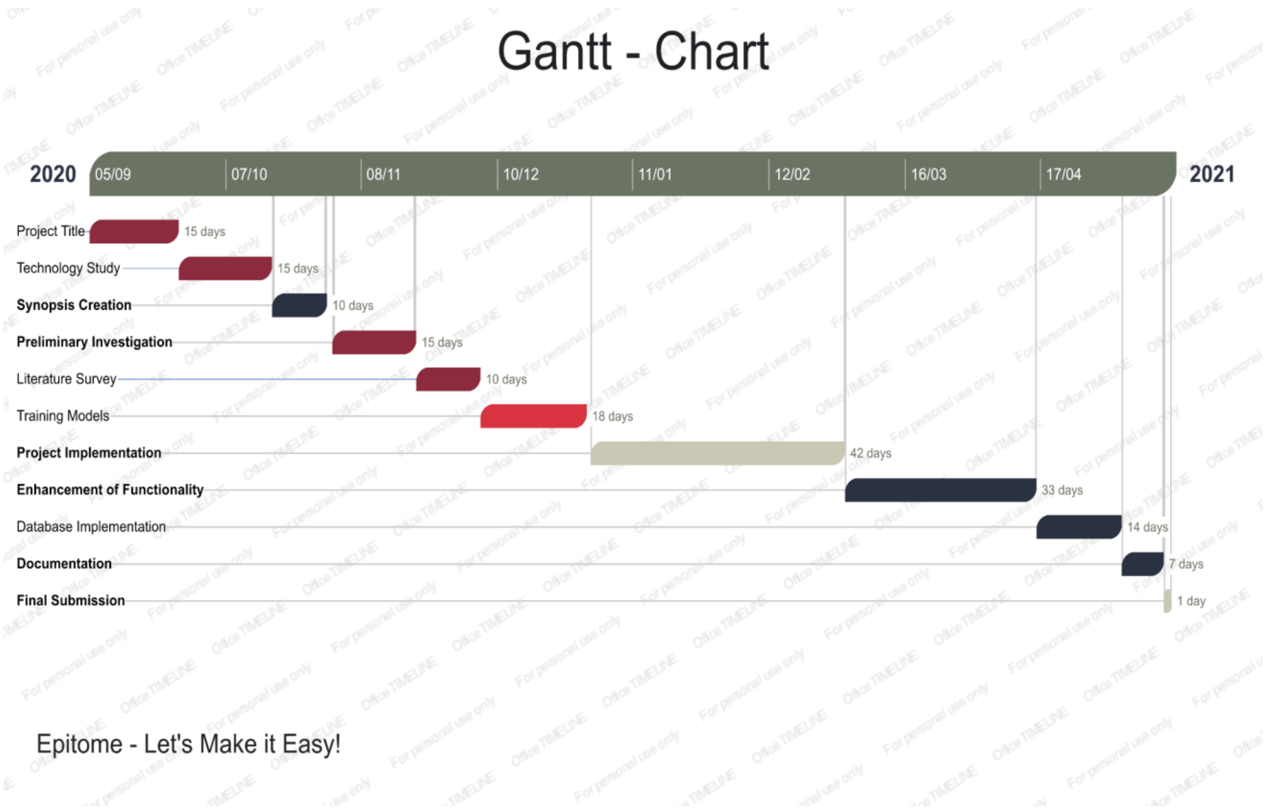
[22] S. Chaudhary, "Extractive Text Summarization Using Neural Networks", Web Blog Post - heartbeat.fritz.ai, May 2018.

[23] Https://app.diagrams.net

Guide Interaction Sheet

Date	Discussion	Action Plan
5/09/2020	Discussed about the title of the Project	Text Summarization Technique Evaluation and Algorithm selection as per the project.
4/10/2020	Discussion on the technology to be used for Summarization of Text and audio in real-time web conferencing.	Django, NLP, Machine Learning and other tools were finalized.
18/10/2020	Discussion of the creation of synopsis of the project	Gathering of information for synopsis creation
01/11/2020	Suggestions on how to do a literature survey and preliminary investigation on the topic	Many research papers were read , understood and their abstract were to be written.
6/12/2020	Discussion on the implementation of the project	Development of speech-to-text and Summarization algorithm through NLP on Jupyter Note book.
30/12/2020	Discussion on the objective of the project(More Efficient Algorithm to Perform Text-Summarization)	Decided to include more extra- -ction layers to increase the efficiency of the Algorithm.
07/02/2021	Suggestion for Front-end Modules to create user Interactive Interfaces with involving all the functionalities.	User Interactive Front-end is developed with all the functionalities and embedded with backend using API.
14/03/2021	For generation of log files and storing the result, database was advised to be added	Action taken that for each user an entry must be made in the database so that user login can be made easy
18/04/2021	Discussion on project documentation	Decided to write the content and integrate it in the proper format of the report.

Gantt-Chart


Gantt - Chart
2020
05/09
07/10
08/11
10/12
11/01
12/02
16/03
17/04
2021
Project Title
15 days
Technology Study
15 days
Synopsis Creation
10 days
Preliminary Investigation
15 days
Literature Survey
10 days
Training Models
18 days
Project Implementation
42 days
Enhancement of Functionality
33 days
Database Implementation
14 days
Documentation
7 days
Final Submission
1 day
Epitome - Let's Make it Easy!

Source Code

settings.py

```
"""
Django settings for myproject project.

Generated by 'django-admin startproject' using Django
3.2.3.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

from pathlib import Path
import os


# Build paths inside the project like this: BASE_DIR /
'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for
production
# See
https://docs.djangoproject.com/en/3.2/howto/deployment/ch
ecklist/

# SECURITY WARNING: keep the secret key used in
production secret!
SECRET_KEY = 'django-insecure-
pn!phmo*506!hy34oxm0xwa$376u2&tmrqy3_a&tv8ig$!)$xe'

# SECURITY WARNING: don't run with debug turned on in
production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'epitome.apps.EpitomeConfig',
    'users.apps.UsersConfig',
    'crispy_forms',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware
',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddlew
are',
    'django.contrib.messages.middleware.MessageMiddleware
',
    'django.middleware.clickjacking.XFrameOptionsMiddlewa
re',
]

ROOT_URLCONF = 'myproject.urls'

TEMPLATES = [
    {
        'BACKEND':
'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug
',
                'django.template.context_processors.reque
st',
                'django.contrib.auth.context_processors.a
uth',
                'django.contrib.messages.context_processo
rs.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'myproject.wsgi.application'


# Database
```

```
#
https://docs.djangoproject.com/en/3.2/ref/settings/#datab
ases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
#
https://docs.djangoproject.com/en/3.2/ref/settings/#auth-
password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSim
ilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthVal
idator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordVa
lidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordV
alidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True
```

```
USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-
files/

STATIC_URL = '/static/'

MEDIA_ROOT=os.path.join(BASE_DIR,'media')
MEDIA_URL='/media/'

# Default primary key field type
#
https://docs.djangoproject.com/en/3.2/ref/settings/#defau
lt-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

CRISPY_TEMPLATE_PACK = 'bootstrap4'

LOGIN_REDIRECT_URL='epitome-home'

LOGIN_URL='login'

EMAIL_BACKEND =
'django.core.mail.backends.console.EmailBackend'
```

url.py (Main)

```
from django.contrib import admin
from django.contrib.auth import views as auth_views
from django.urls import path,include
from users import views as user_views
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('register/',
user_views.register,name='register'),
    path('audio/', user_views.Audio_store,name='audio'),
    path('contact/', user_views.contact,name='epitome-
contact'),
    path('profile/', user_views.profile,name='profile'),
    path('login/',
auth_views.LoginView.as_view(template_name='users/login.h
tml'),name='login'),
    path('logout/',
auth_views.LogoutView.as_view(template_name='users/logout
.html'),name='logout'),
    path('', include('epitome.urls')),
]

if settings.DEBUG:
    urlpatterns+=static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

url.py (epitome)

```
from django.urls import path
from . import views
from .views import
AddLiveAudio,PostListView,PostDetailView,PostCreateView,P
ostUpdateView, PostDeleteView, TextToSummary, AudioToText

urlpatterns = [
    path('', views.first, name='epitome-first'),
    path('team/', views.team, name='epitome-team'),
    path('contact/', views.contact, name='epitome-
contact'),
    path('post/', PostListView.as_view(), name='epitome-
home'),
    path('post/<int:pk>/', PostDetailView.as_view(),
name='post-detail'),
    path('post/<int:pk>/text', AudioToText.as_view(),
name='epitome-text'),
    path('post/<int:pk>/summary',
TextToSummary.as_view(), name='epitome-summary'),
    path('post/new/', PostCreateView.as_view(),
name='post-create'),
    path('post/<int:pk>/live', AddLiveAudio.as_view(),
name='epitome-live'),
    path('post/<int:pk>/update/',
PostUpdateView.as_view(), name='post-update'),
    path('post/<int:pk>/delete/',
PostDeleteView.as_view(), name='post-delete'),
    path('about/', views.about, name='epitome-about'),
]
```

views.py (epitome)

```
from django.shortcuts import render
from django.http import HttpResponse
from django.contrib.auth.mixins import
LoginRequiredMixin, UserPassesTestMixin
from .models import Post
from django.shortcuts import redirect

from django.views.generic import (
    ListView,
    DetailView,
    CreateView,
    UpdateView,
    DeleteView
)

print('Hello')
#current_user = request.user
#print(current_user)
#print(Post.objects.filter(author='Sarthak'))

# Create your views here.

def first(request):
    return render(request, 'epitome/first.html')

def home(request):
    current_user = request.user
    print(current_user)
    print(Post.objects.filter(author='Sarthak'))
    context = {
        'posts': Post.objects.all()
        #'posts': User.objects.filter(author='Sarthak')
    }
    #print(Post.objects.get('author'='Sarthak'))
    print(context)
    print (current_user)
    return render(request, 'epitome/home.html', context)

def team(request):
    return render(request, 'epitome/team.html')

def contact(request):
   return render(request, 'epitome/contact.html')

class PostListView(ListView, LoginRequiredMixin,
UserPassesTestMixin):
    #print(Post.objects.filter(author='Sarthak'))
    model = Post
    template_name = 'epitome/home.html'  #
<app>/<model>_<viewtype>.html
    context_object_name = 'posts'

    #def get_Text(self):
     #   post1=Post.objects.filter(author=self.request.us
er)
      #  print("sdassas",post1)
```

```
        # return Post.objects.all()

    #ordering = ['-date_posted']
    def get_queryset(self):
        #print("Hiiii",self.request.user)
        return
Post.objects.filter(author=self.request.user).order_by('-
date_posted')

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.author:
            return True
        return False



class AudioToText(DetailView, LoginRequiredMixin,
UserPassesTestMixin):

    model = Post

    def get(self, request,*args, **kwargs):
        q=Post.objects.get(id=self.get_object().id).recor
d.name
        q='media/'+q
        print(q)
        path = q
        text = get_large_audio_transcription(path)
        print("\nFull text:",text)
        Post.objects.filter(id=self.get_object().id).upda
te(content=text)
        #return
Post.objects.filter(id=self.get_object().id)
        return
redirect('/post/'+str(self.get_object().id))

    def test_func(self):
        get_data()
        post = self.get_object()
        if self.request.user == post.author:
            print("xyz")
            return True
        return False

class TextToSummary(DetailView, LoginRequiredMixin,
UserPassesTestMixin):

    model = Post

    def get(self, request,*args, **kwargs):
        p=Post.objects.get(id=self.get_object().id).conte
nt
        text=p
        #text = file.read()
        #print("Content:")
        print(text)
        tokenized_sentence = sent_tokenize(text)
        #print(tokenized_sentence)
        text = remove_special_characters(str(text))
        text = re.sub(r'\d+', '', text)
        #print(text)

        tokenized_words_with_stopwords =
word_tokenize(text)
        #print(tokenized_words_with_stopwords)
        tokenized_words = [word for word in
tokenized_words_with_stopwords if word not in Stopwords]
        tokenized_words = [word for word in
tokenized_words if len(word) > 1]
        tokenized_words = [word.lower() for word in
tokenized_words]
        tokenized_words =
lemmatize_words(tokenized_words)
        #print(tokenized_words)
        word_freq = freq(tokenized_words)
        input_user = int(30)
        #input_user = int(input('Percentage of
information to retain(in percent):'))
        no_of_sentences = int((input_user *
len(tokenized_sentence))/100)
        #print(len(tokenized_sentence))
        #print(no_of_sentences)
        c = 1
        sentence_with_importance = {}
        for sent in tokenized_sentence:
            sentenceimp =
sentence_importance(sent,word_freq,tokenized_sentence)
            sentence_with_importance[c] = sentenceimp
            c = c+1

        #print(sentence_with_importance)
        sentence_with_importance =
sorted(sentence_with_importance.items(),
key=operator.itemgetter(1),reverse=True)
        cnt = 0
        summary = []
        sentence_no = []
        for word_prob in sentence_with_importance:
            if cnt < no_of_sentences:
                sentence_no.append(word_prob[0])
                cnt = cnt+1
            else:
                break
        sentence_no.sort()
        cnt = 1
        for sentence in tokenized_sentence:
            if cnt in sentence_no:
                summary.append(sentence)
            cnt = cnt+1
        summary = " ".join(summary)
        #print("\n")
        print("Summary:")
        print(summary)
        Post.objects.filter(id=self.get_object().id).upda
te(summary=summary)
        #Post.objects.filter(content=p).update(summary=su
mmary)
        #Post.objects.all().update(summary='summary')
        #outF = open('summary.txt',"w")
        #outF.write(summary)
        return
redirect('/post/'+str(self.get_object().id))

    def test_func(self):
        get_data()
        post = self.get_object()
        if self.request.user == post.author:
            print("xyz")
            return True
        return False

class PostDetailView(DetailView, LoginRequiredMixin,
UserPassesTestMixin):
    model = Post

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.author:
            print("xyz")
            return True
        return False

class PostCreateView(LoginRequiredMixin, CreateView):
    model = Post
    fields = ['title', 'record']

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)

class PostUpdateView(LoginRequiredMixin,
UserPassesTestMixin, UpdateView):
    model = Post
    fields = ['title', 'record', 'content']

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.author:
            return True
        return False

class PostDeleteView(LoginRequiredMixin,
UserPassesTestMixin, DeleteView):
    model = Post
    success_url = '/post/'

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.author:
            return True
        return False

def about(request):
    return render(request, 'epitome/about.html',
{'title': 'About'})


import os
import speech_recognition as sr
from pydub import AudioSegment
from pydub.silence import split_on_silence

#record=Audio_store.objects.filter('')
#print(record)

r = sr.Recognizer()
# a function that splits the audio file into chunks
# and applies speech recognition
def get_large_audio_transcription(path):
    """
    Splitting the large audio file into chunks
    and apply speech recognition on each of these chunks
    """
    # open the audio file using pydub
    sound = AudioSegment.from_wav(path)
    # split audio sound where silence is 700 miliseconds
or more and get chunks
    chunks = split_on_silence(sound,
        # experiment with this value for your target
audio file
        min_silence_len = 500,
        # adjust this per requirement
        silence_thresh = sound.dBFS-14,
        # keep the silence for 1 second, adjustable as
well
        keep_silence=500,
    )
    folder_name = "audio-chunks"
    # create a directory to store the audio chunks
    if not os.path.isdir(folder_name):
        os.mkdir(folder_name)
    whole_text = ""
    # process each chunk
    for i, audio_chunk in enumerate(chunks, start=1):
        # export audio chunk and save it in
        # the `folder_name` directory.
        chunk_filename = os.path.join(folder_name,
f"chunk{i}.wav")
        audio_chunk.export(chunk_filename, format="wav")
        # recognize the chunk
        with sr.AudioFile(chunk_filename) as source:
            audio_listened = r.record(source)
            # try converting it to text
            try:
                text = r.recognize_google(audio_listened)
            except sr.UnknownValueError as e:
                print("Error:", str(e))
            else:
                text = f"{text.capitalize()}. "
                print(chunk_filename, ":", text)
                whole_text += text
    # return the text for all chunks detected
    return whole_text

import nltk
import os
import re
import math
import operator
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize,word_tokenize

#nltk.download('averaged_perceptron_tagger')
#nltk.download('stopwords')
#nltk.download('punkt')
#nltk.download('wordnet')
Stopwords = set(stopwords.words('english'))
wordlemmatizer = WordNetLemmatizer()
```

```
def lemmatize_words(words):
    lemmatized_words = []
    for word in words:
       lemmatized_words.append(wordlemmatizer.lemmatize(w
ord))
    return lemmatized_words

def stem_words(words):
    stemmed_words = []
    for word in words:
       stemmed_words.append(stemmer.stem(word))
    return stemmed_words

def remove_special_characters(text):
    regex = r'[^a-zA-Z0-9\s]'
    text = re.sub(regex,'',text)
    return text

def freq(words):
    words = [word.lower() for word in words]
    dict_freq = {}
    words_unique = []
    for word in words:
       if word not in words_unique:
           words_unique.append(word)
    for word in words_unique:
       dict_freq[word] = words.count(word)
    return dict_freq

def pos_tagging(text):
    pos_tag = nltk.pos_tag(text.split())
    pos_tagged_noun_verb = []
    for word,tag in pos_tag:
        if tag == "NN" or tag == "NNP" or tag == "NNS" or
tag == "VB" or tag == "VBD" or tag == "VBG" or tag ==
"VBN" or tag == "VBP" or tag == "VBZ":
             pos_tagged_noun_verb.append(word)
    return pos_tagged_noun_verb

def tf_score(word,sentence):
    freq_sum = 0
    word_frequency_in_sentence = 0
    len_sentence = len(sentence)
    for word_in_sentence in sentence.split():
        if word == word_in_sentence:
            word_frequency_in_sentence =
word_frequency_in_sentence + 1
    tf =  word_frequency_in_sentence/ len_sentence
    return tf
```

```
def idf_score(no_of_sentences,word,sentences):
    no_of_sentence_containing_word = 0
    for sentence in sentences:
        sentence =
remove_special_characters(str(sentence))
        sentence = re.sub(r'\d+', '', sentence)
        sentence = sentence.split()
        sentence = [word for word in sentence if
word.lower() not in Stopwords and len(word)>1]
        sentence = [word.lower() for word in sentence]
        sentence = [wordlemmatizer.lemmatize(word) for
word in sentence]
        if word in sentence:
            no_of_sentence_containing_word =
no_of_sentence_containing_word + 1
    idf =
math.log10(no_of_sentences/no_of_sentence_containing_word
)
    return idf

def tf_idf_score(tf,idf):
    return tf*idf

def word_tfidf(dict_freq,word,sentences,sentence):
    word_tfidf = []
    tf = tf_score(word,sentence)
    idf = idf_score(len(sentences),word,sentences)
    tf_idf = tf_idf_score(tf,idf)
    return tf_idf

def sentence_importance(sentence,dict_freq,sentences):
     sentence_score = 0
     sentence = remove_special_characters(str(sentence))
     sentence = re.sub(r'\d+', '', sentence)
     pos_tagged_sentence = []
     no_of_sentences = len(sentences)
     pos_tagged_sentence = pos_tagging(sentence)
     for word in pos_tagged_sentence:
          if word.lower() not in Stopwords and word not
in Stopwords and len(word)>1:
                word = word.lower()
                word = wordlemmatizer.lemmatize(word)
                sentence_score = sentence_score +
word_tfidf(dict_freq,word,sentences,sentence)
     return sentence_score

# import required libraries
import sounddevice as sd
from scipy.io.wavfile import write
import wavio as wv
```

```
'''def Live(request):
    # Sampling frequency
    freq = 44100

    # Recording duration
    duration = 5

    # Start recorder with the given values
    # of duration and sample frequency
    print('StartRecording')
    recording = sd.rec(int(duration * freq),
                    samplerate=freq, channels=2)

    # Record audio for the given number of seconds
    sd.wait()
    print('StopRecording')


    # This will convert the NumPy array to an audio
    # file with the given sampling frequency
    write("media/recordings/recording0.wav", freq,
recording)

    # Convert the NumPy array to audio file
   # wv.write("recording1.wav", recording, freq,
sampwidth=2)'''

class AddLiveAudio(DetailView, LoginRequiredMixin,
UserPassesTestMixin):

    model = Post

    def get(self, request,*args, **kwargs):
        freq = 44100
        duration = 5
        print('StartRecording')
        recording = sd.rec(int(duration * freq),
                        samplerate=freq, channels=2)

        sd.wait()
        print('StopRecording')
        write("media/recordings/recordingst.wav", freq,
recording)
        wv.write("media/recordings/recording_"+str(self.g
et_object().id)+".wav", recording, freq, sampwidth=2)

        p=Post.objects.get(id=self.get_object().id).recor
d
        print(p)
        Post.objects.filter(id=self.get_object().id).upda
te(record='recordings/recording_'+str(self.get_object().i
d)+'.wav')
        #p=Post.objects.get(id=self.get_object().id)
        #p.record='recordings/recording0.wav'
        #p.save()

        print(Post.objects.get(id=self.get_object().id).r
ecord)
        return
redirect('/post/'+str(self.get_object().id))

    '''def get_context_data(self, *args, **kwargs):
        #context =
super(AudioToText,self).get_context_data(*args, **kwargs)
        #context["id"]=self.object.id
        #context["summary"]="xyzzzzzz"
        #context["category"] = "MISC"

        #p=Post.objects.values_list('record', flat =
True).filter(id=self.object.id)
        q=Post.objects.get(id=self.object.id).record.name
        #p=context["record"]
        # val=context['id']
        q='media/'+q
        print(q)
        path = q
        text = get_large_audio_transcription(path)
        print("\nFull text:",text)
        context["summary"]=text
        Post.objects.filter(id=self.object.id).update(con
tent=text)
        return context'''

    def test_func(self):
        get_data()
        post = self.get_object()
        if self.request.user == post.author:
            print("xyz")
            return True
        return False
```

views.py (users)

```
from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from .forms import UserRegisterForm, UserUpdateForm,
ProfileUpdateForm, AudioForm, ContactForm
from epitome.models import Post
from .models import Audio_store
from django.core.mail import send_mail, BadHeaderError
from django.http import HttpResponse

# Create your views here.

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Your account has
been created! You are now able to log in')
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html',
{'form': form})

@login_required
def profile(request):
    if request.method == 'POST':
        u_form = UserUpdateForm(request.POST,
instance=request.user)
        p_form = ProfileUpdateForm(request.POST,
                                   request.FILES,
                                   instance=request.user.
profile)
        if u_form.is_valid() and p_form.is_valid():
            u_form.save()
            p_form.save()
            messages.success(request, f'Your account has
been updated!')
            return redirect('profile')

    else:
        u_form = UserUpdateForm(instance=request.user)
        p_form =
ProfileUpdateForm(instance=request.user.profile)

    context = {
        'u_form': u_form,
        'p_form': p_form
    }

    return render(request, 'users/profile.html', context)

'''def Audio_store(request):
    form = AudioForm()
    return render(request, 'aud.htm', {'form' : form})'''
def Audio_store(request):
    if request.method == 'POST':
        print("In post block")
        form = AudioForm(request.POST,request.FILES or
None)
        if form.is_valid():
            form.save()
            return HttpResponse('successfully uploaded')
    else:
        form = AudioForm()
    return render(request, 'users/aud.html', {'form' :
form})

def contact(request):
    if request.method == 'POST':
        form = ContactForm(request.POST)
        if form.is_valid():
            subject = "Website Inquiry"
            body = {
            'first_name':
form.cleaned_data['first_name'],
            'last_name': form.cleaned_data['last_name'],
            'email': form.cleaned_data['email_address'],
            'message':form.cleaned_data['message'],
            }
            message = "\n".join(body.values())

            try:
                send_mail(subject, message,
'admin@example.com', ['admin@example.com'])
            except BadHeaderError:
                return HttpResponse('Invalid header
found.')
            return redirect ('/about/')

    form = ContactForm()
    return render(request, "epitome/contact.html",
{'form':form})
```

<u>Templates</u>

base.html

```
{% load static %}
<!DOCTYPE html>
<html>
<head>

    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1, shrink-to-fit=no">
```

```
    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css
/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/d
AiS6JXm" crossorigin="anonymous">

    <link rel="stylesheet" type="text/css" href="{%
static 'epitome/main.css' %}">

    {% if title %}
        <title>Epitome - {{ title }}</title>
    {% else %}
        <title>Epitome</title>
    {% endif %}
</head>
<body>
    <header class="site-header">
      <nav class="navbar navbar-expand-md navbar-dark bg-
steel fixed-top">
        <div class="container">
          <a class="navbar-brand mr-4" href="{% url
'epitome-first' %}">Epitome</a>
          <button class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbarToggle" aria-
controls="navbarToggle" aria-expanded="false" aria-
label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
          </button>
          <div class="collapse navbar-collapse"
id="navbarToggle">
            <div class="navbar-nav mr-auto">
              <a class="nav-item nav-link" href="{% url
'epitome-first' %}">Home</a>
              {% if user.is_authenticated %}
                <a class="nav-item nav-link" href="{% url
'epitome-home' %}">Posts</a>
              {% endif %}
              <a class="nav-item nav-link" href="{% url
'epitome-about' %}">About</a>
              <a class="nav-item nav-link" href="{% url
'epitome-team' %}">Team</a>
              <a class="nav-item nav-link" href="{% url
'epitome-contact' %}">Contact Us</a>


            </div>
            <!-- Navbar Right Side -->
            <div class="navbar-nav">
              {% if user.is_authenticated %}
```

```
                <a class="nav-item nav-link" href="{% url
'post-create' %}">New Post</a>
                <a class="nav-item nav-link" href="{% url
'profile' %}">Profile</a>
                <a class="nav-item nav-link">Welcome {{
user.username }}!</a>
                <a class="nav-item nav-link" href="{% url
'logout' %}">Logout</a>

              {% else %}
                <a class="nav-item nav-link" href="{% url
'login' %}">Login</a>
                <a class="nav-item nav-link" href="{% url
'register' %}">Register</a>
              {% endif %}
            </div>
          </div>
        </div>
      </nav>
    </header>
    <main role="main" class="container">
      <div class="row">
        <div class="col-md-12">
          {% if messages %}
            {% for message in messages %}
              <div class="alert alert-{{ message.tags
}}">
                {{ message }}
              </div>
            {% endfor %}
          {% endif %}
          {% block content %}{% endblock %}
        </div>

        <!---<div class="col-md-4">
          <div class="content-section">
            <h3>Our Sidebar</h3>
            <p class='text-muted'>You can put any
information here you'd like.
              <ul class="list-group">
                <li class="list-group-item list-group-
item-light">Latest Posts</li>
                <li class="list-group-item list-group-
item-light">Announcements</li>
                <li class="list-group-item list-group-
item-light">Calendars</li>
                <li class="list-group-item list-group-
item-light">etc</li>
              </ul>
            </p>
          </div>
        </div>--->
      </div>
    </main>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS
-->
    <script src="https://code.jquery.com/jquery-
3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93h
XpG5KkN" crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.1
2.9/umd/popper.min.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXus
vfa0b4Q" crossorigin="anonymous"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/b
ootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+7
6PVCmYl" crossorigin="anonymous"></script>
</body>
</html>
```

first.html

```
{% extends "epitome/base.html" %}
{% block content %}
  <section class='col-md-12 border-style: dotted;'>
    <h3 class='text-center'>Welcome to Epitome! Let's
make it easy...</h3>
    <div class="row">
      <div class="col-md-1 mt-5" style="text-align:
justify;"> </div>
      <div class="col-md-4 " style="margin-top: 100px;
text-align: justify;">

        <p>Epitome is one of those applications of
Natural Language Processing (NLP)
          which is bound to have a huge impact on our
lives. With growing digital media
          and ever growing publishing - who has the time
to go through entire articles
           / documents / books to decide whether they are
useful or not?</p>
        </div>
        <div class="col-md-1 mt-5" style="text-align:
justify;"> </div>
        <div class="col-md-6">
          <img style="max-width: 90%; height: auto;"
src="media/image.jpg">
```

```
          </div>
      </div>
  </section>
{% endblock content %}
```

home.html (posts)

```
{% extends "epitome/base.html" %}
{% block content %}
  {% if posts|length > 0 %}
  <h3 class='text-center'>Posts</h3>
    {% for post in posts %}
        <article class="media content-section">
          <img class="rounded-circle article-img" src="{{
post.author.profile.image.url }}">
            <div class="media-body">
              <div class="article-metadata">
                <a class="mr-2" href="#">{{ post.author
}}</a>
                <small class="text-muted">{{
post.date_posted|date:"F d, Y" }}</small>
              </div>
              <h2><a class="article-title" href="{% url
'post-detail' post.id %}">{{ post.title }}</a></h2>
              <p class="article-content" style="text-
align: justify;"><b>Content:</b> {{ post.content }}</p>
              <p class="article-content" style="text-
align: justify;"><b>Summary:</b> {{ post.summary }}</p>
            </div>
        </article>
    {% endfor %}
  {% else %}
  <article class="media content-section">
      <div class="media-body">
        <h2>No Posts</a></h2>
      </div>
  </article>
  {% endif %}
{% endblock content %}
```

contact.html

```
{% extends "epitome/base.html" %}
{% load crispy_forms_tags %}
{% block content %}
  <section class='col-md-12 border-style: dotted;'>
    <div class="row">
      <div class="col-md-6 mt-4" style="text-align:
justify;">
```

```
            <h3>Contact Us</h3>
        <h4>Contact us directly if you have any
questions</h4>
        <p>
          Please write your name, email address and a
message below if you have any questions.
          One of our staff members will be happy to
contact you directly and answer your questions as soon as
possible.
        </p>
        <form method="post">
            {% csrf_token %}
                {{form|crispy}}
                <button type="submit">Submit</button>
        </form>
        </div>
        <div class="col-md-6">
          {% load static %}
            <img style="margin-top: 200px; max-width:
100%; height: auto;" alt='Image' src="{% static
'images/group.jpg' %}">
          </div>
      </div>
  </section>
{% endblock content %}
```

about.html

```
{% extends "epitome/base.html" %}
{% block content %}
    <section class='col-md-12 border-style: dotted;'>
        <h3 class='text-center'>About</h3>
        <div class="row">
            <div class="col-md-1 mt-5" style="text-align:
justify;"> </div>
        <div class="col-md-4 " style="margin-top: 70px;
text-align: justify;">
            <p>The project is to create a product which
joins Google, Zoom, and other web conferencing meetings.
While one is in the online/offline meeting speaking,
Epitome is recording the audio of the conversation. Once
the meeting concludes, the Epitome natural language
processing (NLP) algorithms transcribe the audio into a
copy of the text and skim it to provide the summary (key
content) of the meeting rather than full transcription.
            </p>
            </div>
            <div class="col-md-1 mt-5" style="text-align:
justify;"> </div>
            <div class="col-md-6">
            {% load static %}
            <img style="max-width: 90%; height: auto;"
alt='Image' src="{% static 'images/image.jpg' %}">
            </div>
        </div>
    </section>
{% endblock content %}
```

team.html

```
{% extends "epitome/base.html" %}
{% block content %}
  <section id="team">
    <div class="container">
      <h3 class="title1 text-center">Project Team</h3>
      <div class="row text-center">
        <div class="col-md-4 team">
          {% load static %}
            <img style="max-width: 90%; height: auto;"
alt='Image' src="{% static 'images/sarthakp (2).jpg'  %}"
class="team-img">
          <p><b>Sarthak Parakh</b><br>Team
Member<br>"Ingenious & Sophisticated"</p>
        </div>
        <div class="col-md-4 team">
          {% load static %}
            <img style="max-width: 60%; height: 50%;"
alt='Image' src="{% static 'images/photo.jpg'  %}"
class="team-img">
          <p><b>Shivam Goyan</b><br>Team
Member<br>"Passionate & Agile"</p>
        </div>
        <div class="col-md-4 mt-5">
          {% load static %}
            <img style="max-width: auto; height: 40%;"
alt='Image' src="{% static 'images/somya.jpg'  %}"
class="team-img">
          <p><b>Soumya Jain</b><br>Team
Member<br>"Constructive & Sincere"</p>
        </div>
    </div>
  </section>
{% endblock content %}
```

post_form.html

```
{% extends "epitome/base.html" %}
{% load crispy_forms_tags %}
{% block content %}
    <div class="content-section">
        <form enctype="multipart/form-data"
method="POST">
```

```
            {% csrf_token %}
            <fieldset class="form-group">
                <legend class="border-bottom mb-
4">Summary Post</legend>
                {{ form|crispy }}
            </fieldset>
            <div class="form-group">
                <button class="btn btn-outline-info"
type="submit">Post</button>
            </div>
        </form>
    </div>
{% endblock content %}
```

post_detail.html

```
{% extends "epitome/base.html" %}
{% block content %}
  <article class="media content-section">
    <img class="rounded-circle article-img" src="{{
object.author.profile.image.url }}">
      <div class="media-body">
        <div class="article-metadata">
          <a class="mr-2" href="#">{{ object.author
}}</a>
          <small class="text-muted">{{
object.date_posted|date:"F d, Y" }}</small>
          {% if object.author == user %}
          <div>
            <a class="btn btn-secondary btn-sm mt-1 mb-1"
href="{% url 'post-update' object.id %}">Update</a>
            <a class="btn btn-secondary btn-sm mt-1 mb-1"
href="{% url 'epitome-text' object.id %}">Audio To
Text</a>
            <a class="btn btn-secondary btn-sm mt-1 mb-1"
href="{% url 'epitome-summary' object.id %}">Text To
Summary</a>
            <a class="btn btn-secondary btn-sm mt-1 mb-1"
href="{% url 'epitome-live' object.id %}">Record Live</a>
            <a class="btn btn-danger btn-sm mt-1 mb-1"
href="{% url 'post-delete' object.id %}">Delete</a>
          </div>
        {% endif %}
        </div>
        <h2 class="article-title">{{ object.title }}</h2>
        <p class="article-content" style="text-align:
justify;"><b>Recording File Name:</b> {{ object.record
}}</p>
        <p class="article-content" style="text-align:
justify;"><b>Content:</b> {{ object.content }}</p>
```

```
        <p class="article-content" style="text-align:
justify;"><b>Summary:</b> {{ object.summary }}</p>

      </div>
  </article>
{% endblock content %}
```

post_confirm_delete.html

```
{% extends "epitome/base.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST">
            {% csrf_token %}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Delete
Post</legend>
                <h2>Are you sure you want to delete the
post "{{ object.title }}"</h2>
            </fieldset>
            <div class="form-group">
                <button class="btn btn-outline-danger"
type="submit">Yes, Delete</button>
                <a class="btn btn-outline-secondary"
href="{% url 'post-detail' object.id %}">Cancel</a>
            </div>
        </form>
    </div>
{% endblock content %}
```

login.html

```
{% extends "epitome/base.html" %}
{% load crispy_forms_tags %}
{% block content %}
<section class='col-md-12 mt-4 border-style: dotted;'>
    <div class="row">
        <div class=" col-md-8 text-center">
            {% load static %}
            <img style="max-width: 70%; height: auto;"
alt='Image' src="{% static 'images/image.jpg' %}">

        </div>

        <div class="content-section col-md-4">
            <form method="POST">
                {% csrf_token %}
                <fieldset class="form-group">
                    <legend class="border-bottom mb-
4">Log In</legend>
                    {{ form|crispy }}
                </fieldset>
                <div class="form-group">
                    <button class="btn btn-outline-info"
type="submit">Login</button>
                </div>
            </form>
            <div class="border-top pt-3">
                <small class="text-muted">
                    Need An Account? <a class="ml-2"
href="{% url 'register' %}">Sign Up Now</a>
                </small>
            </div>
        </div>
</section>
{% endblock content %}
```

logout.html

```
{% extends "epitome/base.html" %}
{% block content %}
    <h2>You have been logged out</h2>
    <div class="border-top pt-3">
        <small class="text-muted">
           <a href="{% url 'login' %}">Log In Again</a>
        </small>
    </div>
{% endblock content %}
```

profile.html

```
{% extends "epitome/base.html" %}
{% load crispy_forms_tags %}
{% block content %}
    <div class="content-section">
      <div class="media">
        <img class="rounded-circle account-img"
alt="Image of Seal" src="{{ user.profile.image.url }}">
        <div class="media-body">
          <h2 class="account-heading">{{ user.username
}}</h2>
          <p class="text-secondary">{{ user.email }}</p>
        </div>
      </div>
      <!-- FORM HERE -->
      <form method="POST" enctype="multipart/form-data">
        {% csrf_token %}
        <fieldset class="form-group">
            <legend class="border-bottom mb-4">Profile
Info</legend>
            {{ u_form|crispy }}
            {{ p_form|crispy }}
        </fieldset>
        <div class="form-group">
            <button class="btn btn-outline-info"
type="submit">Update</button>
        </div>
    </form>
    </div>
{% endblock content %}
```

register.html

```
{% extends "epitome/base.html" %}
{% load crispy_forms_tags %}
{% block content %}
<section class='col-md-12 mt-4 border-style: dotted;'>
    <div class="row">
        <div class="text-center col-md-8">
            {% load static %}
            <img style="max-width: 70%; height: auto;"
alt='Image' src="{% static 'images/image.jpg' %}">

        </div>

    <div class="content-section col-md-4">
        <form method="POST">
            {% csrf_token %}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Join
Today</legend>
                {{ form|crispy }}
            </fieldset>
            <div class="form-group">
                <button class="btn btn-outline-info"
type="submit">Sign Up</button>
            </div>
        </form>
        <div class="border-top pt-3">
            <small class="text-muted">
                Already Have An Account? <a class="ml-2"
href="{% url 'login' %}">Sign In</a>
            </small>
        </div>
    </div>
</section>


{% endblock content %}
```

Models

models.py (epitome)

```
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User
from django.urls import reverse

# Create your models here.

class Post(models.Model):
    title = 
models.CharField(max_length=100,default='Title')
    record = 
models.FileField(default='default.wav',upload_to='recordi
ngs/')
    content = models.TextField(null=True, blank=True)
    summary = models.TextField(null=True, blank=True)
    date_posted = 
models.DateTimeField(default=timezone.now)
    author = models.ForeignKey(User, 
on_delete=models.CASCADE)

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse('post-detail', kwargs={'pk': 
self.pk})
```

model.py (users)

```
from django.db import models
from django.contrib.auth.models import User
from PIL import Image

# Create your models here.

class Profile(models.Model):
    user = models.OneToOneField(User, 
on_delete=models.CASCADE)
    image = models.ImageField(default='default.jpg', 
upload_to='profile_pics')

    def __str__(self):
        return f'{self.user.username} Profile'

    def save(self, **kwargs):
        super().save()

        img = Image.open(self.image.path)
        if img.height > 300 or img.width > 300:
            output_size = (300, 300)
            img.thumbnail(output_size)
            img.save(self.image.path)
```