BLOG MINING AND EMOTION ARGUMENTATION

A PROJECT REPORT

Submitted in partial fulfillment of the requirementsfor the award of the degree of

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

Submitted by

MANDALAPU SRUTHI
18BQ1A1288
KODURU SARASWATHI
18BQ1A1263
KONAKANCHI VEENA MADHURI
18BQ1A1268
MANNAVA SRAVYA
18BQ1A1293

Under the Supervision of
Ms. Vineela Kanakamedala, M.Tech

Assistant Professor

VVIT
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY
Jawaharlal Nehru Technological University, Kakinada, AP, India

JUNE 2022

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY : : NAMBUR

BONAFIDE CERTIFICATE

This is to certify that this project report **"Blog Mining and Emotion Argumentation"** is the bonafide work of "**M. SRUTHI (18BQ1A1288) K. SARASWATHI (18BQ1A1263) K. VEENA MADHURI (18BQ1A1268) M. SRAVYA (18BQ1A1293)**", who carried out the project under the supervision of **"*Ms. Vineela Kanakamedala*"** academic year 2021-22 towards partial fulfillment of the Degree of Bachelor of Technology in Information Technology from Jawaharlal Nehru Technological University, Kakinada. The results embodied in this report have not been submitted to any other University for the award of any degree.

Signature of the Head of the Department	Signature of the Supervisor
Dr. A. Kalavathi	**Ms. K. Vineela**
HEAD OF THE DEPARTMENT	**GUIDE**
Department of INFORMATION TECHNOLOGY	Assistant Prof. INFORMATION TECHNOLOGY N Block, VVIT

External Viva voce conducted on _____

Internal Examiner **External Examiner**

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY :: NAMBUR

CERTIFICATE OF AUTHENTICATION

I solemnly declare that this project report **"Blog Mining and Emotion Argumentation"** is the bonafide work done totally by us, carried out under the supervision of **Ms. K. Vineela**, towards partial fulfillment of the requirements of the Degree of Bachelor of Technology in Information Technology from Jawaharlal Nehru Technological University, Kakinada during the year 2021-22.

It is further certified that this work has NOT been submitted, either in part or in full, to any other department of Jawaharlal Nehru Technological University, or any other University, Institution or elsewhere, or for publication in any form.

Signature of the Student

DATE:

MANDALAPU SRUTHI
18BQ1A1288, 2021-22

KODURU SARASWATHI
18BQ1A1263, 2021-22

KONAKANCHI VEENA MADHURI
18BQ1A1268, 2021-22

MANNAVA SRAVYA
18BQ1A1293, 2021-22

ACKNOWLEDGEMENT

We take this opportunity to express our deepest gratitude and appreciation to all those people who made this project work easier with words of encouragement, motivation, discipline, and faith by offering different places to look to expand my ideas and helped me towards the successful completion of this project work.

First and foremost, we express our deep gratitude to **Mr. Vasireddy Vidya Sagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the Information Technology program.

We express our sincere thanks to **Dr. Y. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the Information Technology program.

We express our sincere gratitude to **Dr. A. Kalavathi**, Professor & HOD, Information Technology, Vasireddy Venkatadri Institute of Technology for her constant encouragement, motivation and faith by offering different places to look to expand my ideas. We would like to express our sincere gratefulness to our guide **Ms. K. Vineela**, MTech, and Project Coordinator, **Md. Shakeel Ahmed**, Associate Professor, further insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project.

We would like to take this opportunity to express our thanks to the teaching and non-teaching staff in Department of Information Technology,

VVIT for their invaluable help and support.

Table of Contents

CHAPTER III SYSTEM DESIGN

CHAPTER IV SYSTEM IMPLEMEMTATION

CHAPTER V SYSTEM TESTING

CHAPTER VI CONCLUSION

CHAPTER VII Appendices

LIST OF FIGURES

CHAPTER I INTRODUCTION

1.1 INTRODUCTION

A blog (a shortened version of "weblog") is an online journal or informational website displaying information in reverse chronological order, with the latest posts appearing first, at the top. It is an informational website, often informal diary-style text entries, and acting as a platform where a writer or a group of writers share their views on an individual subject. It is similar to an online journal where an individual, group, or corporation presents a record of activities, thoughts, or beliefs. In general terminology blog is explained as a regularly updated website or web page, typically run by an individual or small group, that is written in an informal or conversational style, consisting of series of posts where posts are archived, and are usually sorted into categories. It is similar to a newspaper in that it publishes new items on a regular basis and keeps the older ones up to date. Blogs are often updated webpages that provide important information about a specific topic. Food, fashion, and marketing are just a few of the topics covered by various sorts of blogs. The purpose of a blog is to provide content on your website that answers prospective customers questions and helps them learn about your product or service. It expands visibility by giving Google and other search engines content to index and serve up in search results. Blogging is a part where there is collection of skills that one needs to run and supervise a blog. This entails equipping a web page with tools to make the process of writing, posting, linking, and sharing content easier on the internet. Bloggers identify the sentiments, both positive and negative opinions about the topic to understand and present public views in detail. Readers can browse these categories through the blog to read older entries. It does typically involve searching and analyzing blogs in order to generate additional insights and acts as an information source for the user's ideas. Blog Mining and Emotion Argumentation provides a capability of processing large amount of text data effectively, blog mining can be a valuable method for gaining insights into a given topic. It is the process of exploring the contents and analyzing blogs in order to generate additional insights that might not be found by examining a single blog. Due to its capability of processing large amount of text data effectively, blog mining can be a valuable method for gaining insights into a given topic. This study of blog mining is used to analyze and search the online blog posts relevant contents in a quite simpler fashion.

1.2 AIM AND SCOPE

Blog mining allows to extract the huge amount of corpus and allows to view it in as condensed state. An individual user diverges respective contents of blog, the encapsulated content allows to digest the huge info in a compact way, consuming minimal time impact. The processing of contents via browsing reduces the actual time consumption to lower. The series of blogs when updated via from the end of the user, which there by provides a summarized view updation to end of database. The main constraints involved are blog corpus source, depictive info of a blog and representative contents, Machine Learning view impact to overcome the complication.

1.3 GOALS AND OBJECTIVES

The typical goals of the Blog mining are to collect the blog corpus and design a system for topic identification and other text processing tasks such as text summarization unit, text categorization, and information routing. In inclusion argumentation mining occupies a position between natural language processing, argumentation theory and information retrieval. It does aims for automatic detect, classify and structure argumentation in text. It focuses on the detection of all the arguments in a text and their relationships with their preceding and following arguments. As the amount of text keeps growing, it becomes increasing difficult for humans to process the deluge of information in the time available. It does result in consumption of huge amount of duration. As a part of low efficiency, the subject propagated within the blogs may not be reachable to end users. Blog Mining overcomes by performing text routing techniques like text summarization. The overloaded content which is extracted from huge number of posts provides low search results, as a process of application of summarization providing faster pace of search.

1.4 DESCRIPTION OF EXISTING SYSTEM

Blogs are perhaps widely used by the internet users due to their ability to disseminate information and present their ideas on various topics. These blogs have increasingly become an important information source for the users' ideas, sensitivity and sentiments. These subjective information in blogs helps in understanding a blogger's views and observations about various topics. All posts are archived in blog, and are usually sorted into categories. Readers can browse these categories of a blog to read older entries. As the overloaded content which are extracted from huge number of posts might even result low search results.

1.5 DESCRIPTION OF PROPOSED SYSTEM

Blog Mining and Emotion Argumentation Project is to mine the information available in the Blogs, one of the important communicative and informative repositories of text based emotional contents in the Web. As the amount of on-line text keeps growing, it becomes increasing difficult for humans to process the deluge of information in the time available. Automatic text processing is an obvious solution to the information overload problem in blog mining. We need automatic text processing systems to help us scan through huge volume of texts, route them to relevant parties, filter them into prespecified categories, or even summarize them. The proposed system pulls out search optimization and allows users to acknowledge deluge information of blogging websites. It explains about keyword research which involves finding the keywords (or search queries) that are extracted from the blog. It emphasizes with Machine learning based text processing tasks such as text summarization unit, text categorization, and information routing. Involvement of text summarization allows to primarily produce tokens, where emphasized tokens are recognized, through which arguments are assigned for preceding and following tokens. This process allows consumption time to be shorter having quicker search results.

1.6 FEASIBILITY STUDY

Preliminary investigation examines project feasibility, the study is a high-level capsule version of the entire process intended to answer a number of questions like: What is the problem? Is there any feasible solution to the given problem? Is the problem even worth solving? Feasibility study is conducted once the problem is clearly understood. Feasibility study is necessary to determine that the proposed system is Feasible by considering the technical, Operational, and Economical factors. By having a detailed feasibility study the management will have a clear-cut view of the proposed system. A well-designed feasibility study should provide a historical background of the business or project, the operations and management, marketing research and policies, financial data, legal requirements and tax obligations. The following feasibilities are considered for the project in order to ensure that the project is variable and it does not have any major obstructions. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation: Technical feasibility, Operational feasibility, Economical feasibility.

1.6.1 TECHNICAL FEASIBILITY

Technical feasibility is the formal process of assessing whether it is technically possible to manufacture a product or service. In this step, we verify whether the proposed systems are technically feasible or not. i.e., all the technologies required to develop the system are available readily or not. Technical Feasibility determines whether the organization has the technology and skills necessary to carry out the project and how this should be obtained. It is one of the first studies that must be conducted after the project has been identified. Technical feasibility study includes the hardware and software devices. The technical issue usually raised during the feasibility stage of the investigation includesthe following: Does the necessary technology exist to do what is suggested, Do the proposed equipment's have the technical capacity to hold the data required to use the newsystem, Will the proposed system provide adequate response to inquiries, regardless of the number or locationof users, Can the system be upgraded if developed, Are there technical guarantees of accuracy, reliability, ease of access and data security, Does system is flexible and it can be expanded further, Does technology needed for project is readily available.

1.6.2 OPERATIONAL FEASIBILITY

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. Proposed projects are beneficial only if they deploy it in internet, then we can take the complete advantage of the project. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. we verify different operational factors of the proposed systems like manpower, time etc., whichever solution uses less operational resources, is the best operationally feasible solution. The solution should also be operationally possible to implement. Operational Feasibility is a measure of how well a proposed system solves the problem and takes advantage of the opportunities identified during scope definition. It determines if the proposed system satisfied user objectives could be fitted into the current system operation. Some of the important issues raised are to test the operational feasibility of a project includes the following: Solve the problem blogger security, quick results, quality search results, the proposed system will not cause any problem under any circumstances.

1.6.3 ECONOMICAL FEASIBILITY

The purpose of economic feasibility is to determine the positive economic benefits that include quantification and identification. This device will help people to save their time. As there will be no wastage of time, the user will be satisfied. It will also help the clearance of traffic in an efficient manner. Assigning two minutes of time for one vehicle is not encouraged in this project. Time is allocated as per the traffic only. The system is economically feasible due to availability of all requirements such as collection of data from blog. We could get daily visitors for the blog, that improves the traffic of the website. By this traffic we can run advertising different in your blog, that will generate the income for us. We will get a good brand value; we can collect a subscription from users to read the content from the bloggers.

CHAPTER II REQUIREMENT SPECIFICATION

2.1 DESCRIPTION OF A PROBLEM

Bloggers are facing their toughest challenge yet and it's called content saturation. There's just too much information to process these days. And when people feel overwhelmed, they react in ways that aren't good for a respective blog. Information overload occurs when a person is exposed to more information than the brain can process at one time. As the amount of on-line text keeps growing, it becomes increasing difficult for humans to process the deluge of information in the time available. The rapid growth of blog documents in web and categorizing search applications based on topics motivates to develop a system that provides identification of the blog documents.

2.2 PROPOSED SOLUTION

Text summarization is the process of creating a short, accurate, and fluent summary of a longer text document. Automatic text summarization methods are greatly needed to address the ever-growing amount of text data available online to both better help discover relevant information and to consume relevant information faster. There is an enormous amount of textual material, and it is only growing every single day where textual information in the form of digital documents quickly accumulates to huge amounts of data. Think of the internet, comprised of web pages, news articles, status updates, blogs and so much more. The data is unstructured and the best that we can do to navigate it is to use search and skim the results. There is a great need to reduce much of this text data to shorter, focused summaries that capture the salient details, both so we can navigate it more effectively as well as check whether the larger documents contain the information that we are looking for. The amount of data available online is limitless. Think about a normal college student, who has to go through thousands of pages of documents each semester. The main purpose of text summarization is to get the most precise and useful information from a large document and eliminate the irrelevant or less important ones. The process brings out information that is crucial, and also ensures that the meaning of the paragraph stays the same. This helps reduce the time to understand large papers like research articles, without skipping any vital information. The main benefits are makes reading easier, saves time, helps to memorize the information easily, boosts the work rate efficiency. This process allows consumption time to be shorter having quicker search results.

2.3 SYSTEM REQUIREMENTS

System requirements are the configuration that a system must have in order for a hardware or software application to run smoothly and efficiently. System requirements are the required specifications a device must have in order to use certain hardware or software. System requirements are clearly articulated statements of what a system must be able to do in order to satisfy needs and requirements and are derived from business requirements and user requirements.

2.3.1 USER INTERFACES

The user interface (UI) is the point of human-computer interaction and communication in a device. The goal of effective UI is to make the user's experience easy and intuitive, requiring minimum effort on the user's part to receive the maximum desired outcome. UI is created in layers of interaction that appeal to the human senses (sight, touch, auditory and more). They include both input devices like a keyboard, mouse, trackpad, microphone, touch screen, fingerprint scanner, e-pen and camera, and output devices like monitors, speakers and printers. Devices that interact with multiple senses are called "multimedia user interfaces." For example, everyday UI uses a combination of tactile input (keyboard and mouse) and a visual and auditory output (monitor and speakers).

Requirements:

Html
Css
Bootstrap
Java Script

2.3.2 SOFTWARE INTERFACES

Software interfaces (programming interfaces) are the languages, codes and messages that programs use to communicate with each other and to the hardware. Examples are the Windows, Mac and Linux operating systems, SMTP email, IP network protocols and the software drivers that activate the peripheral devices. In contrast to a user interface, which connects a computer to a person, an application programming interface connects computers or pieces of software to each other. It is not intended to be used directly by a person (the end user) other than a computer

programmer who is incorporating it into the software. An API is often made up of different parts which act as tools or services that are available to the programmer. A program or a programmer that uses one of these parts is said to call that portion of the API. The calls that make up the API are also known as subroutines, methods, requests, or endpoints. An API specification defines these calls, meaning that it explains how to use or implement them.

Requirements:

Python (Flask app)
Jupyter Notebook
MySql
Xampp

2.3.3 COMMUNICATION INTERFACES

Communications Interfaces means the interfaces and protocols that enable software, directories, networks, Operating Systems, network Operating Systems or Web-Based Software installed on one computer (including Personal Computers, servers and Handheld Computing Devices) to Interoperate with the Microsoft Platform Software on another computer including without limitation communications designed to ensure security, authentication or privacy.

Requirements:

Internet, Wireless Networks(Wi-fi)

2.3.3.1 FUNCTIONAL REQUIREMENTS

A Functional Requirement (FR) is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behaviour, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements in Software Engineering are also called Functional Specification**.**

Requirements:

User Access:

1. Blog Home
2. Categories
3. Author
4. Search via blogs
5. Visualize respective post contents

Admin Access:

1. Admin Login
2. Admin Reset
3. Reset Password via Mail
4. Blog_Info, Profile_Info
5. Updation of new blog, Edit any of blog updated

User Access:

Blog site consisting header part, having a small depictive logo followed by the pages to redirect of user's choice which comprises of stories which is the home page of the blogging website, categories which is embodied with the posts on basis of selective type of category, author consisting the information to contact and knew detailed view of author, and the last part which is a search bar. The home page followed by the title of the project which is Blog Mining, followed by the posts which are featured where the ordering comes into view having greater extent of views, and next panel provides all stories arranging in the order of latest updated blogs. The next segment you could observe as view all posts which is a URL to redirect, which redirects to the view_all page where containing header part, footer part and the mid region contains all the posts which are updated ordering latest to be visualized at first. On selection of category respective type of blogs relevant to the particular category type is visualized. If no blogs found on respective category gives a specified message "No posts found in this category, Get Back to Home", the URL get back to home redirects to home page. This view even containing search of contents in form of search via category. Giving a search string provides out outcome in relevant to that particular category. The search via category redirects to new page visualizing the blogs where search string is matched, ordered in basis having more matches

that are found at the top. If found no matches, gives a prompt message that “No matches found in this category, Get Back to Home”, the URL get back to home redirects to home page. Author page follows to be an info regard the admin and few other details. To be in contact with the respective admin, the user can simply type of the message having necessary fields the name of the user, respective mail id, the subject and last the message that he wants to send. On click of submit the user can simply receive a small message via mail depicting “we would be in contact with you sooner!!!”, at the same period the related message info is sent to the respective admin who actually dealing with it. Search via nn submission through the search bar the page redirects to the search list page where user can get through the blogs of provided specification. These blogs are arranged in order where attainment of huge number of matches at the top and followed by next and so on…, if the verified search string doesn’t match with any kind of the blog info, it puts a message “No matches found, Get Back to Home”. Post details is carried out by visualizing the title, category of blog, followed by depictive photo, and content which is summarized part of content, so that user could easily grasp up the content. The last part of home page follows to be a footer having a source to get in touch.

DataBase_View:

Blog_Post: (Table of Contents – stored in database)
Id, Category, Title, Updated_By, Updated_On, Num_Views, Content
Id : Specifying the primary key for unique representation of each blog of type int
Category : Provides the which type of category does blogging post encompasses.
Title : Title of the blog which of type varchar()
Updated_By : Provides the details who have updated the blog, takes up type varchar()
Updated_On : Ensures provision of the date when actually blog is updated, of type date
Num_Views : It is of type int, ensuring number of visits to a particular blog
Content : The detailed description of post takes up a type of varchar()

Url:

127.0.0.1:5000/
127.0.0.1:5000/view_all
127.0.0.1:5000/categories/type_of_category
127.0.0.1:5000/search_category/type_of_category
127.0.0.1:5000/author

127.0.0.1:5000/search
127.0.0.1:5000/post/Id_of_blog

Admin Access:

Login page allows admins to login by provision of valid credentials. Here the credential password accessed is verified after decryption, where password is stored by an encrypted format. Respective messages based on actions are encountered, the actions are given as provision of valid username and password, provision of valid username and invalid password, provision of invalid username and valid password and at last providing both username and password as incorrect. The condition where both username and password are tended to be correct will allow to login to a respective page where admin tend to access the related blog info. All other situations tend to be failed and returns a prompt message that might be incorrect username/password. This page provides URL to redirect to home page as well for resetting of password. Reset necessary explains about if admin had forgot the credentials, he can simply issue forgot password, on click of forgot password, it redirects to reset page asking for mail-id, if the respective mail-id is registered sends a reset link to mail and representing a message "Please check your mail!!!", if at all the mail-id is not registered provides a prompt message "Mail-id is not registered!!". The reset link is sent to the mail, it is valid only for an hour, access of reset link after an hour provides token expired, this page is regulated with new password and confirm new password and a submission button. If the passwords are found to be equal redirects to login page on submission, if not equal gives a note that passwords not equal. Here the password stored is encrypted with a key. Blog info is speculated with the info of the admin logged in, the contents involved are number of posts updated, the list of posts having an edit option and a delete option if a user wants to add a new blog, he can redirect to that respective page by click of update a new post, and towards the top right corner is diverged with Sign Out option redirects to login page. The profile info URL diverges to the admin info regard personal scenario. This page does contain sign out button and information of logged in admin, and a link to diverge to the blog_info page. Update a new blog is the updation of new blogging content. This operation is taken once admins logs in successfully and after redirecting from blog_info page. It starts with the fields which are type of category that the blog is comprised, the title of the blog, and necessary information to be updated, pictographic information to be attached. The last field is the button named as update, on submission the information is inserted to the database on basis of which date the information updated, who have updated the information.

Editing of the blog works in the similar fashion, but the aspect to be considered is updation of content is done rather than insertion.

Fileds_within_the_page:

Category	:	type text and is a required field
Title	:	type text and is a required field
Content	:	type text and is a required field
Upload Image	:	type file and is a required field
Submit	:	It is a button, on click the loaded content is updated

Url:

127.0.0.1:5000/login
127.0.0.1:5000/reset
127.0.0.1:5000/reset_email
127.0.0.1:5000/user_name/blog_info
127.0.0.1:5000/user_name/profile_info
127.0.0.1:5000/update_content/user_name
127.0.0.1:5000/blog_info_edit/ADMIN_titleofblog

2.3.3.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are requirements that are not directly concerned with the specified function delivered by the system. They may relate to emergent system properties such as reliability, response time and store occupancy. Some of the non-functional requirements related with this system are hereby below:

Reliability: Reliability based on this system defines the evaluation result of the system, correct identification of the facial expressions and maximum evaluation rate of the facial expression recognition of any input images.

Compatibility: Supportive on minimal hardware requirements and other versions of operating system.

Ease of Use: The system is simple, user friendly, graphics user interface implemented so any can use this system without any difficulties.

Security: The sensitive information like login credentials should be stored securely. The password is here by encrypted with the algo running behind.
Availability: Intended to available 99.9% of the time.

2.3.4 MODULES

CSV: It implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel.

Random: It is an in-built module of Python which is used to generate random numbers. This module can be used to perform random actions such as generating random numbers, print random a value for a list or string, etc.

NumPy: It is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. The arrange([start,] stop[, step,][, dtype]) which is method in NumPy : Returns an array with evenly spaced elements as per the interval. The interval mentioned is half-opened i.e. [Start, Stop]

Matplotlib: Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They’re typically instruments for reasoning about quantitative information. It is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy.

Matplotlib.Pyplot: Pyplot is a collection of functions in the popular visualization package Matplotlib. Its functions manipulate elements of a figure, such as creating a figure, creating a plotting area, plotting lines, adding plot labels, etc.

Flask: Flask is a web framework, it’s a Python module that lets you develop web applications easily. It was developed by Armin Ronacher, who led a team of international Python enthusiasts called Poocco. It’s having a small and easy-to-extend core: it’s a microframework that doesn’t include an ORM (Object Relational Manager) or such features. It does have many cool features like url routing, template engine. It is a WSGI web app framework.

Render_template: Render_template is used to generate output from template file. Templates, they can have access to the config, request, session, g1 objects, the url_for

and get_flashed_messages() functions. It is typically imported directly from the flask package. It Flask's render_template() helper function to serve an HTML template as the response.

Request: The Flask request (source code) object is critical for building web applications with this web framework. The request context allows you to obtain data sent from the client such as a web browser so that you can appropriately handle generating the response. The current request method is available by using the method attribute. To access form data (data transmitted in a POST or PUT request) you can use the form attribute.

Url_for: To build a URL to a specific function, use the url_for() function. It accepts the name of the function as its first argument and any number of keyword arguments, each corresponding to a variable part of the URL rule. Unknown variable parts are appended to the URL as query parameters.

Redirect: Flask class provides the redirect() function which redirects the user to some specified URL with the specified status code. The syntax to use the redirect() function is Flask.redirect(<location>,<status-code>, <response>)

Flash: In the web applications, there are scenarios where the developer might need to flash the messages to provide feedback to the users for the behaviour of the application in different cases.

Flask_mysqldb, MySql: flask-mysqldb provide mysql connection for flask. Import using an import mysql. connector statement so you can use this module's methods to communicate with the MySQL database. Use the cursor() method of a MySQLConnection object to create a cursor object to perform various SQL operations.

Flask_mail, Mail, Message: A web-based application is often required to have a feature of sending mail to the users/clients. Flask-Mail extension makes it very easy to set up a simple interface with any email server. Mail manages email-messaging requirements. encapsulates an email message as: Message (subject, recipients, body, html, sender, cc, bcc, reply-to, date, charset, extra_headers, mail_options, rcpt_options).

NLTK: The Natural Language Toolkit (NLTK) is a Python package for natural language processing. NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization,

stemming, tagging, parsing, and semantic reasoning, and an active discussion forum.

Nltk.corpus, stopwords: NLTK (Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages. You can find them in the nltk_data directory. A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

Tokenize, sent_tokenize, word_tokenize: Tokenizers divide strings into lists of substrings. Natural language processing is used for building applications such as Text classification, intelligent chatbot, sentimental analysis, language translation, etc. Word_tokenize use the method word_tokenize() to split a sentence into words. The output of word tokenization can be converted to Data Frame for better text understanding in machine learning applications. Sent_tokenize is a submodule for word_tokenize.

2.4 SYSTEM ANALYSIS METHODS

System analysis is a procedure or approach that serves to determine the system’s performance for a given (known) structure of this system. An example may be a typical student project with a given input data which should be made for a defined system structure. The resulting calculation data characterize system outputs. The Unified Modelling Language (UML) is a graphical language for OOAD that gives a standard way to write a software system’s blueprint. It helps to visualize, specify, construct, and document the artifacts of an object-oriented system. It is used to depict the structures and the relationships in a complex system.

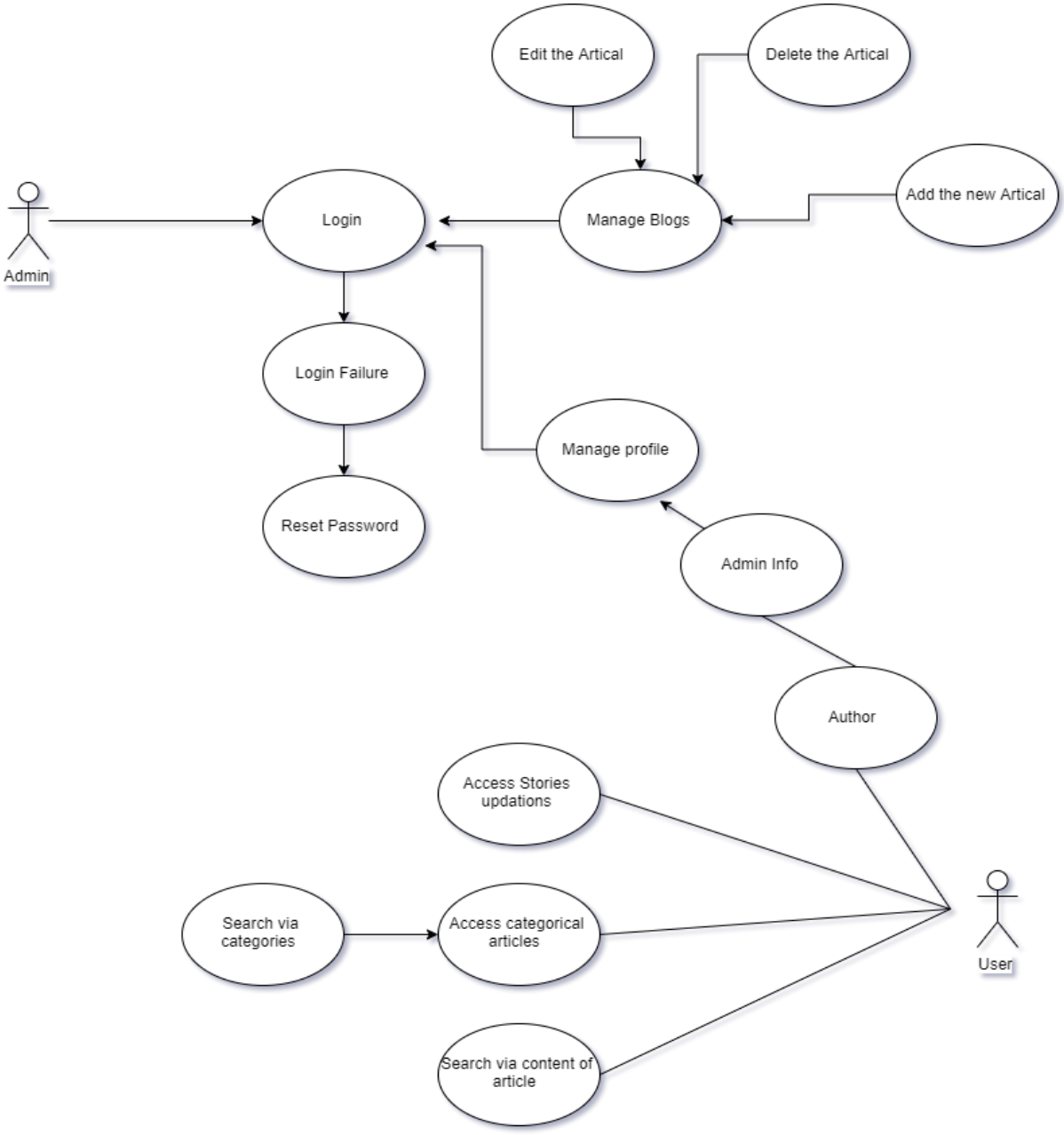
System: A set of elements organized to achieve certain objectives form a system. Systems are often divided into subsystems and described by a set of models.

Model: Model is a simplified, complete, and consistent abstraction of a system, created for better understanding of the system.

View: A view is a projection of a system’s model from a specific perspective.

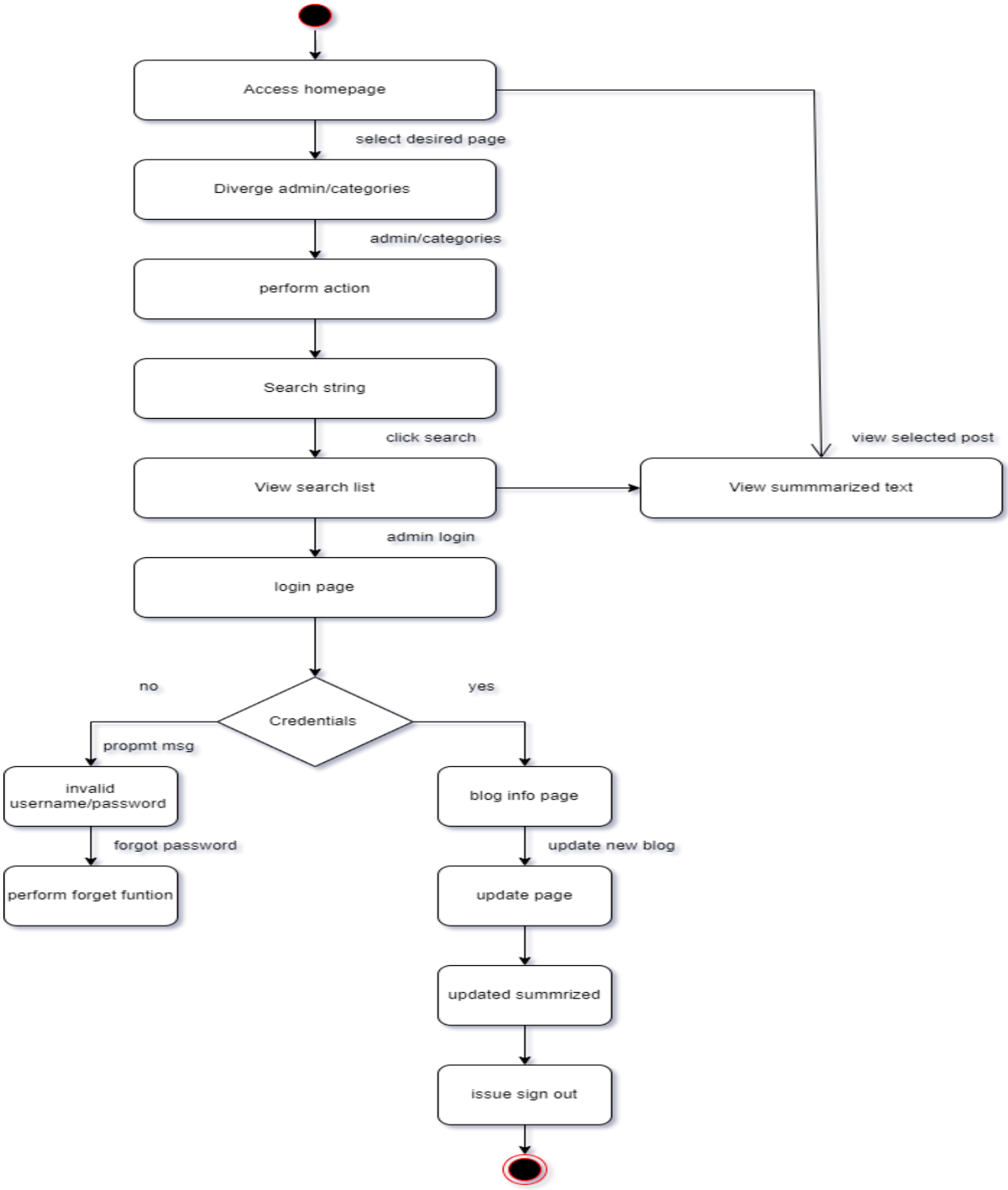
2.4.1 USE CASE DIAGRAM

In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. Common components include: Actors: The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.



2.4.2 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

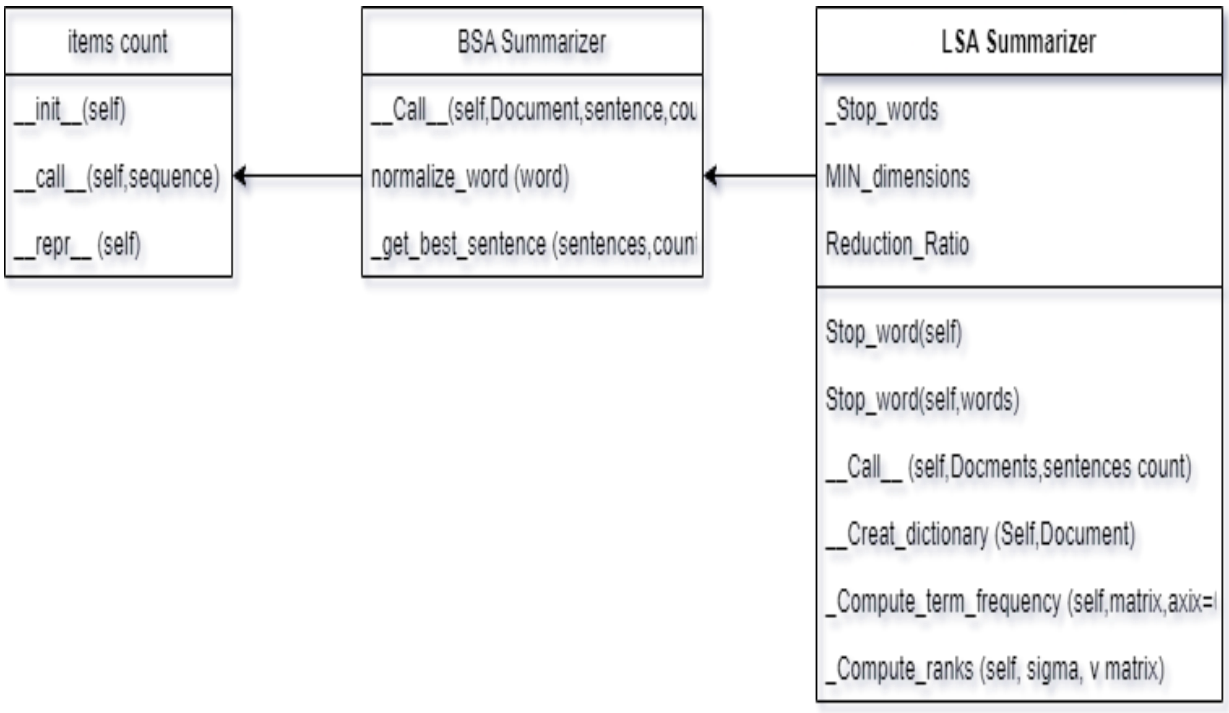


2.5 SYSTEM DESIGN METHODS

Systems design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization. It is a systemic approach is required for a coherent and well-running system. Bottom-Up or Top-Down approach is required to take into account all related variables of the system. A designer uses the modelling languages to express the information and knowledge in a structure of system that is defined by a consistent set of rules and definitions. The designs can be defined in graphical or textual modelling languages.

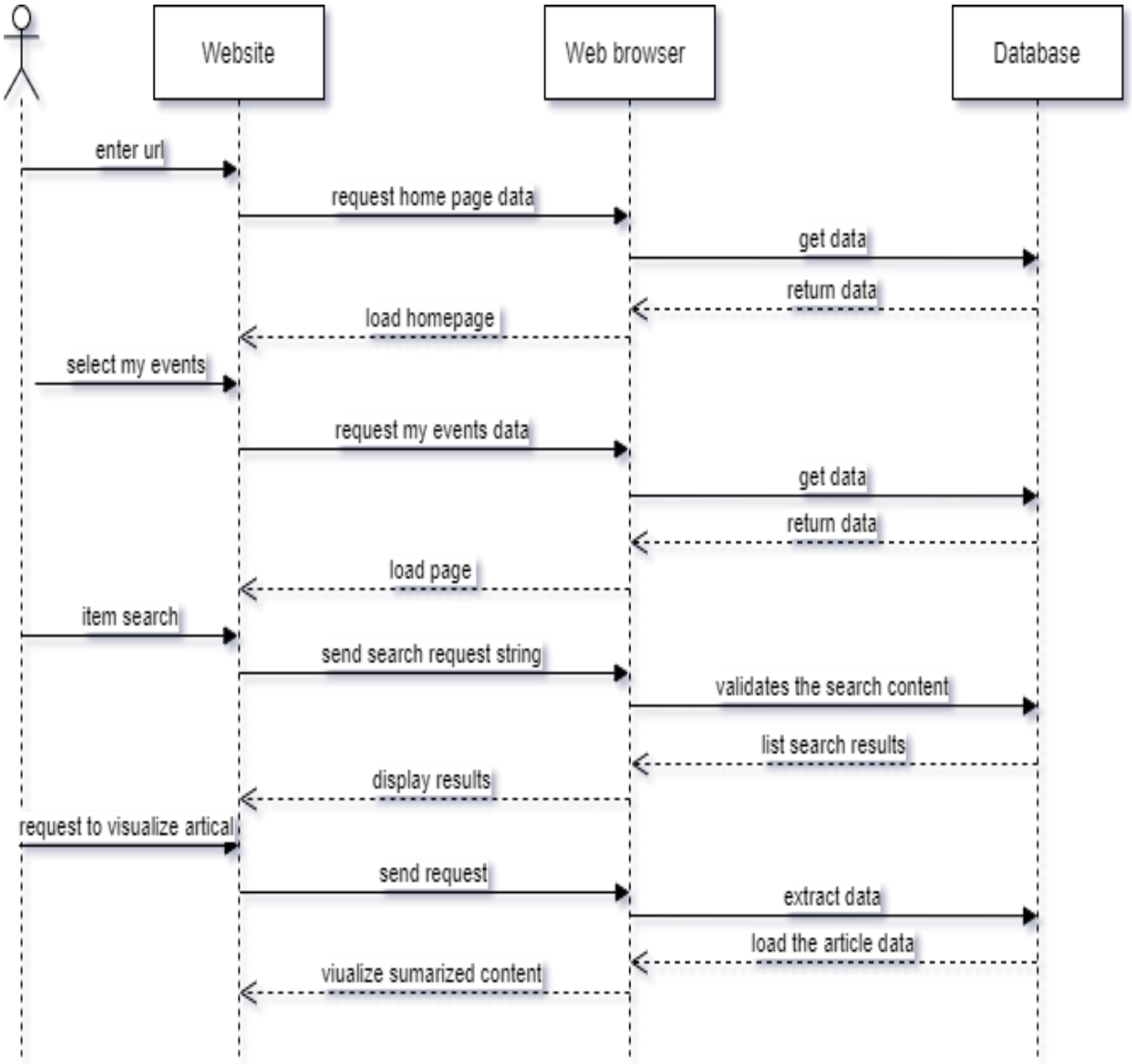
2.5.1 CLASS DIAGRAM

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in theapplication, and the classes to be programmed.



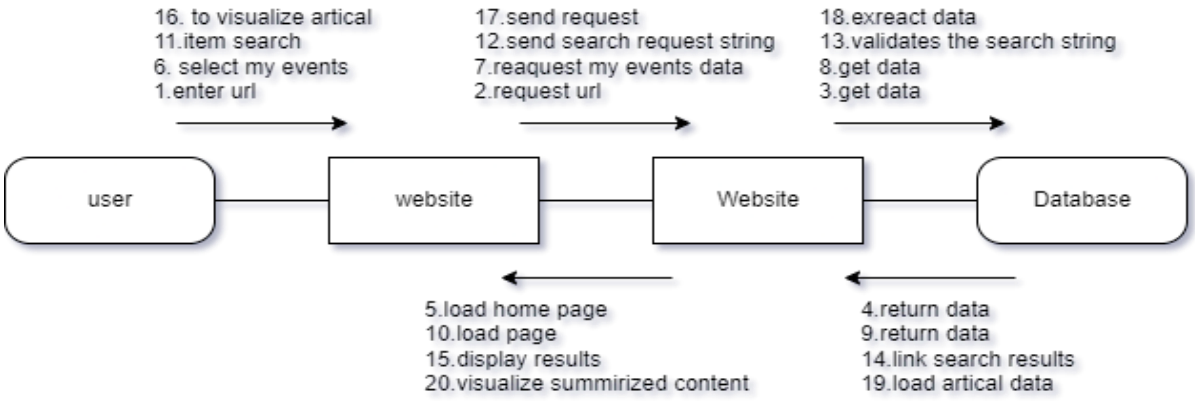
2.5.2 SEQUENCE DIAGRAM

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.



2.5.3 COLLABORATION DIAGRAM

A collaboration diagram resembles a flowchart that portrays the roles, functionalityand behaviors of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with the labels that define the message sequencing.



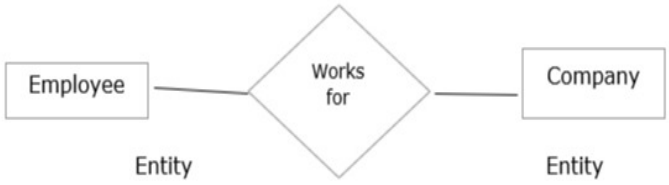
2.6 ENTITY - RELATIONSHIP DIAGRAMS

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

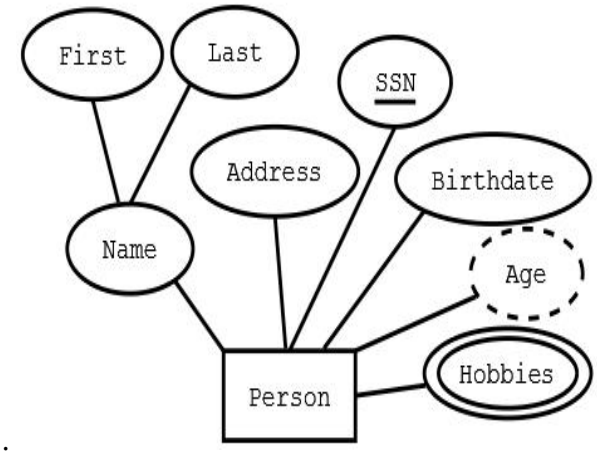
Components of ER diagram:

Entity: It may be an object, person, place or event that stores data in a database. In a relationship diagram an entity is represented in rectangle form. For example, students, employees, managers, etc.

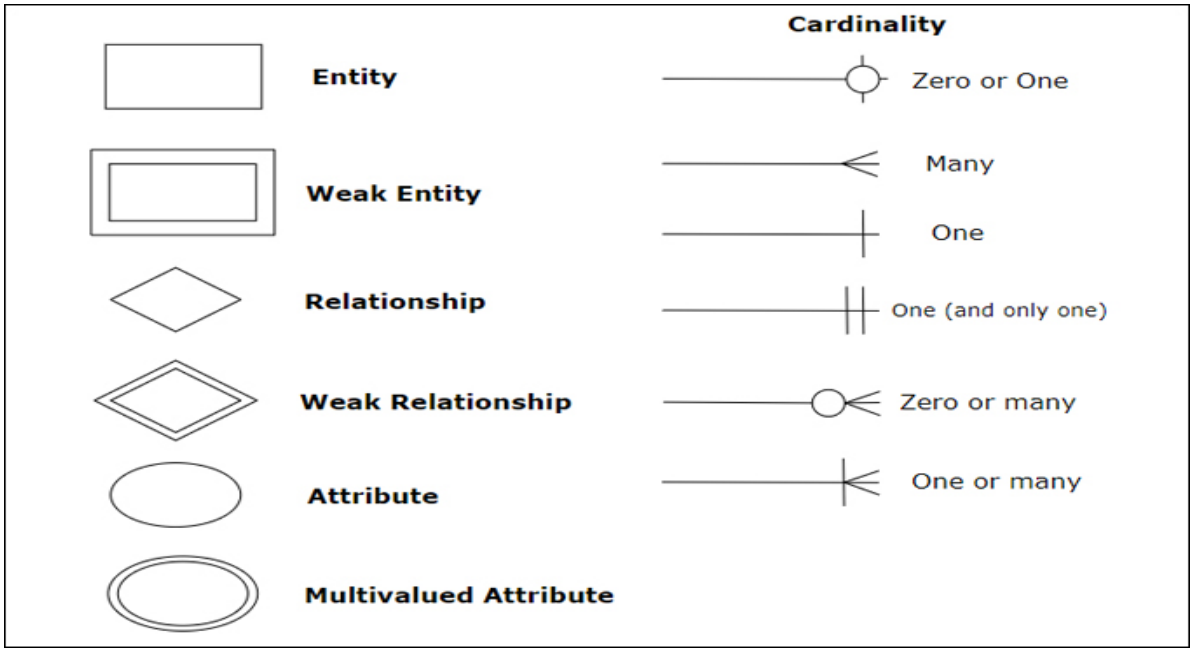
Relationship: It is used to describe the relation between two or more entities. It is represented by a diamond shape.



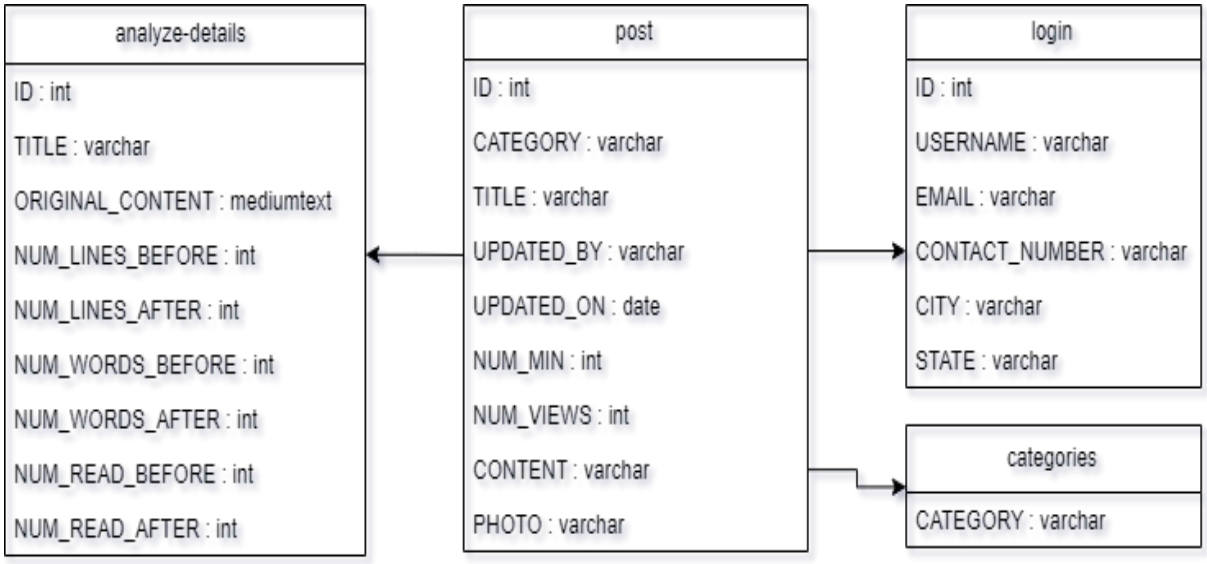
Attributes: It is the name, thing etc. These are the data characteristics of entities or data elements and data fields.



Components:



ER Diagram for Blog Mining:



CHAPTER III SYSTEM DESIGN

3.1 INTRODUCTION

Systems design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization. Systems design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way the process is systematic where in it takes into account all related variables of the system that needs to be created—from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system. Systems design then overlaps with systems analysis, systems engineering and systems architecture.

3.1.1 DESIGN OVERVIEW – ALGORITHM

Algorithm – Latent Semantic Analysis:

The main aim of latent semantic analysis is to create representations of text data in terms of the features and latent features. Latent semantic analysis (LSA) is a mathematical method for computer modelling and simulation of the meaning of words and passages by analysis of representative corpora of natural text. LSA closely approximates many aspects of human language learning and understanding. It supports a variety of applications in information retrieval, educational technology and other pattern recognition problems where complex wholes can be treated as additive functions of component parts. The latent sematic analysis consists of two steps:

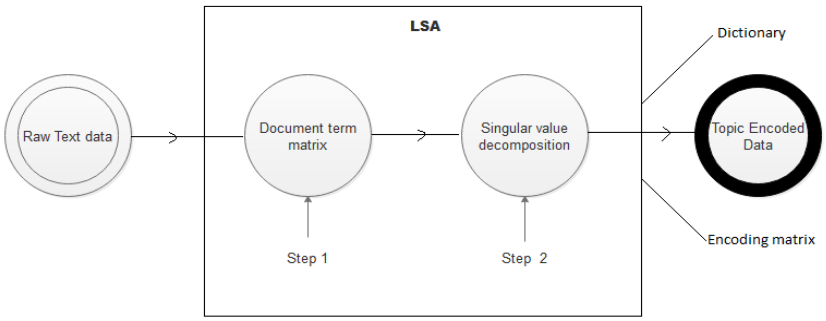
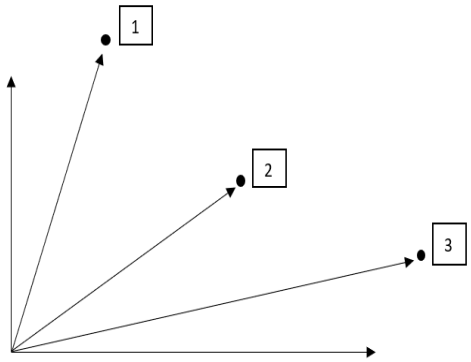


Fig. LSA processing

Step1 : Document Term Matrix

A document-term matrix is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms. This matrix is a specific instance of a document-feature matrix where "features" may refer to other properties of a document besides terms. They are useful in the field of natural language processing and computational text analysis. The basic idea of document term matrix is that text documents can be represented as points in euclidean space:



Point 1: I am document in Euclidean Space
Point 2: Here is another document
Point 3: And here is third

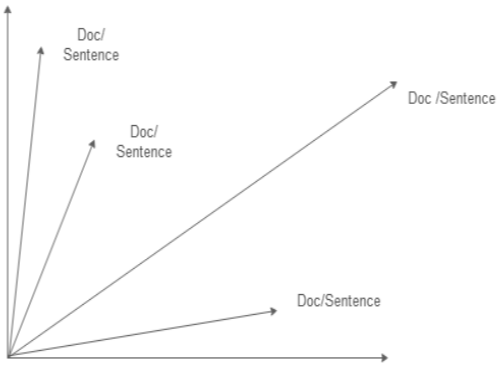


Fig. Euclidean plane

The below is the document term matrix:

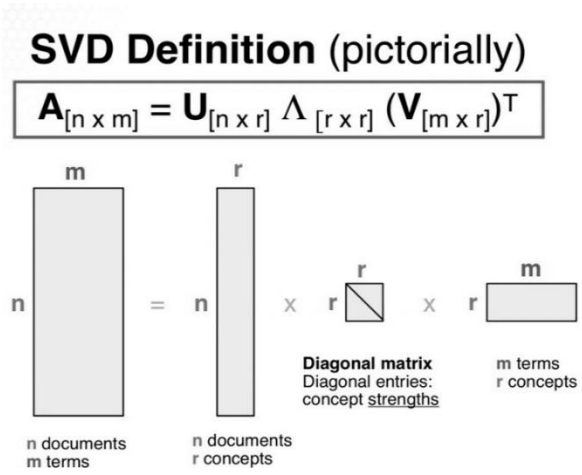
Document Term Matrix

	intelligent	applications	creates	business	processes	bots	are	i	do	intelligence
Doc 1	2	1	1	1	1	0	0	0	0	0
Doc 2	1	1	0	0	0	1	1	0	0	0
Doc 3	0	0	0	1	0	0	0	1	1	1

	brown	dog	fox	lazy	quick	red	slow	the	yellow
“the quick brown fox”	1	0	1	0	1	0	0	1	0
“the slow brown dog”	1	1	0	0	0	0	1	1	0
“the quick red fox”	0	1	0	0	1	1	0	1	0
“the lazy yellow fox”	0	0	1	1	0	0	0	1	1

Step2 : Singular Value Decomposition

It is similar to principal component analysis. It allows to provide the dimensionality of the data set by encoding these with latent features. The Singular Value Decomposition (SVD) of a matrix is a factorization of that matrix into three matrices. It has some interesting algebraic properties and conveys important geometrical and theoretical insights about linear transformations. It also has some important applications in data science. Interpretation of SVD is given by: Deriving mapping between m-dimensional space and r-dimensional singular vector space (rank of input matrix = r). Breaks down the original document into r linearly-independent base vectors or concepts. SVD can semantically cluster words and sentences by finding salient and recurring patterns and representing them by 1 of the singular vectors. The magnitude of the corresponding singular vector indicates the degree of importance of the pattern (salient feature/concept) within the document. The sentence that best represents this pattern will have the largest index value. After you run SVD and get the most salient concepts in the text, we need to select the sentences as summary.



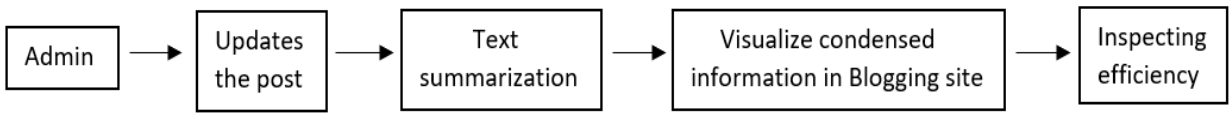
3.2 SYSTEM ARCHITECTURAL DESIGN

A system architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system. A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs). Various organizations can define systems architecture in

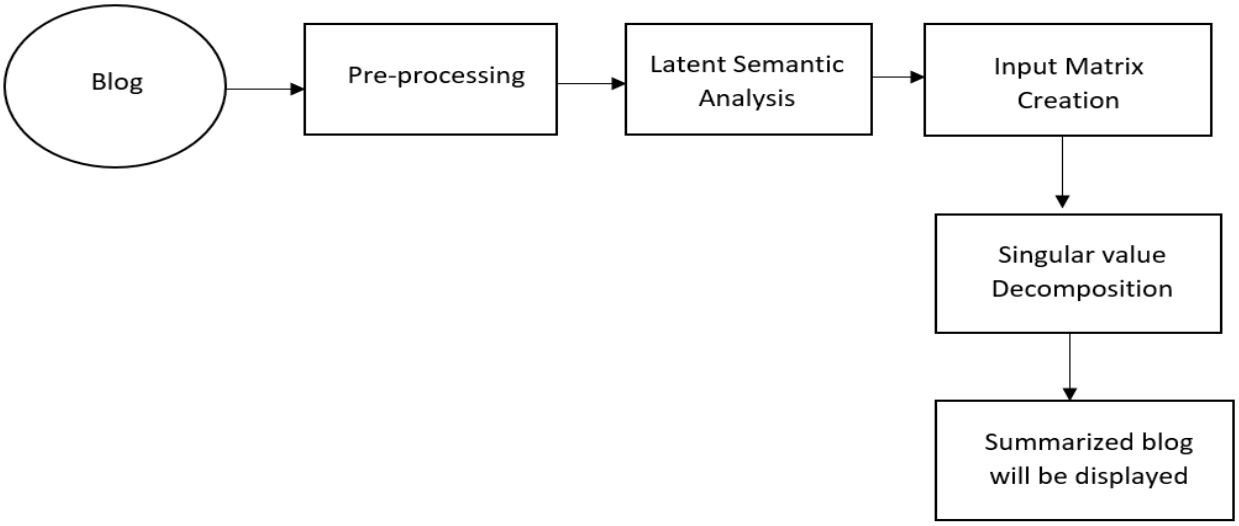
different ways, including: The fundamental organization of a system, embodied in its components, their relationships to each other and to the environment, and the principles governing its design and evolution, A representation of a system, including a mapping of functionality onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components.

3.2.1 CHOOSEN SYSTEM ARCHITECTURE

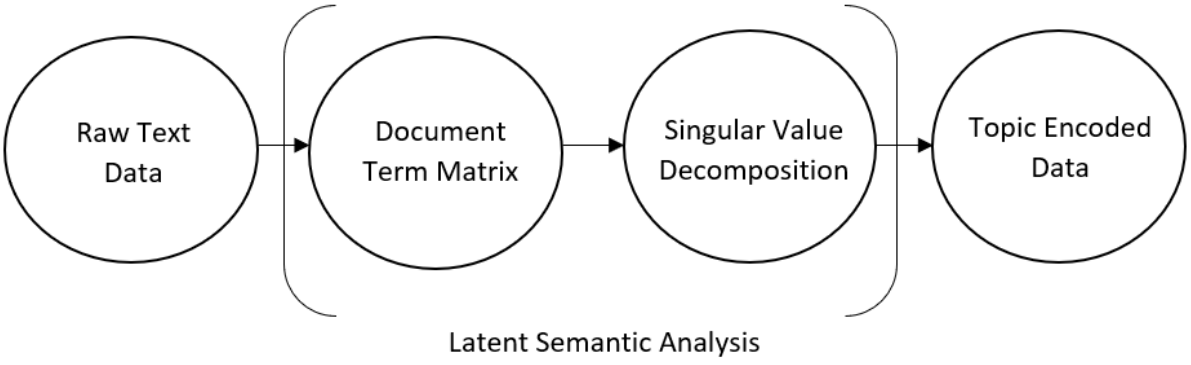
System View:



Text Summarizer:



Latent Semantic Analysis:



3.2.2 SYSTEM INTERFACE DESCRIPTIONS

The blog mining involves admin who updates new blog, here by updated content is summarized in short and is stored into the database. Thus, summarized content is here by accessed by users while access of info regard a blog. This a part where the condensed information in blogging site is visualized. The next part is to inspect efficiency by gather of details regard info before summarization and after summarization. Thus the pre-processed data is allowed to provide it for a graphical representation having a visual impact for comparison of before summarization and after summarization. During this process the tasks involved for text summarizer is blog content is pre-processed applying the algo Latent Semantic Analysis, thus displaying out summarized content of blog. The latent semantic analysis involves typically two steps, first step is document term matrix where the content gathered is represented in a matrix, and then followed by singular value decompose allows to extract the ranked text from the matrix and extracted data which is in encoded matrix format is extracted and pulled out as sentences.

3.3 USER INTERFACE DESIGN

The user interface (UI) is the point of human-computer interaction and communication in a device. The goal of effective UI is to make the user's experience easy and intuitive, requiring minimum effort on the user's part to receive the maximum desired outcome. UI is created in layers of interaction that appeal to the human senses (sight, touch, auditory and more). User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. UI brings together concepts from interaction design, visual design, and information architecture.

3.3.1 DESCRIPTION OF USER INTERFACE

HTML:

Introduction:

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included

cues for the appearance of the document.HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets.

History of HTML:

In 1989, Berners-Lee wrote a memo proposing an Internet-based hypertext system. Berners-Lee specified HTML and wrote the browser and server software in late 1990. That year, Berners-Lee and CERN data systems engineer Robert Cailliau collaborated on a joint request for funding, but the project was not formally adopted by CERN. The first publicly available description of HTML was a document called "HTML Tags", first mentioned on the Internet by Tim Berners-Lee in late 1991.It describes 18 elements comprising the initial, relatively simple design of HTML.

HTML Layout elements:

HTML has several semantic elements that define the different parts of a web page:

<header> : Defines a header for a document or a section

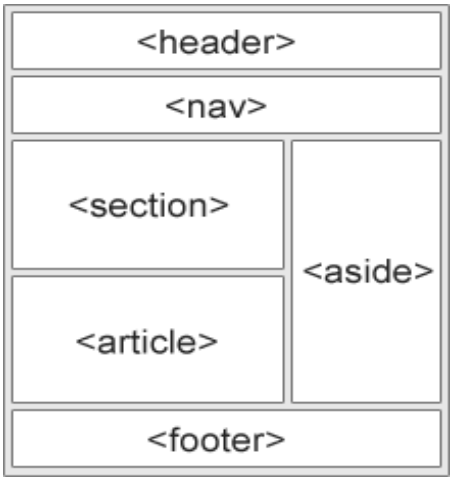
<nav> : Defines a set of navigation links

<section> : Defines a section in a document

<article> : Defines an independent, self-contained content

<aside> : Defines content aside from the content (like a sidebar)

<footer> : Defines a footer for a document or a section.



Basic template of HTML:

```
<!doctype html>
<html>
        <head>
            <title>Our Funky HTML Page</title>
             <meta name="description" content="Our first page">
             <meta name="keywords" content="html tutorial template">
        </head>
        <body>
                <p>content goes in-between these tags</p>
        </body>
</html>
```

CSS:

Introduction:

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable. CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects. CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document.

Types of CSS:

Cascading Style Sheet(CSS) is used to set the style in web pages that contain HTML elements. It sets the background color, font-size, font-family, color, … etc property of elements on a web page. There are three types of CSS which are given below:

Inline CSS : Inline CSS contains the CSS property in the body section attached with element is known as inline CSS. This kind of style is specified within an HTML tag using the style attribute.

Internal CSS : This can be used when a single HTML document must be styled uniquely. The CSS rule set should be within the HTML file in the head section i.e the

CSS is embedded within the HTML file.

External CSS : External CSS contains separate CSS file which contains only style property with the help of tag attributes (For example class, id, heading, … etc). CSS property written in a separate file with .css extension and should be linked to the HTML document using link tag. This means that for each element, style can be set only once and that will be applied across web pages.

CSS Box Model:

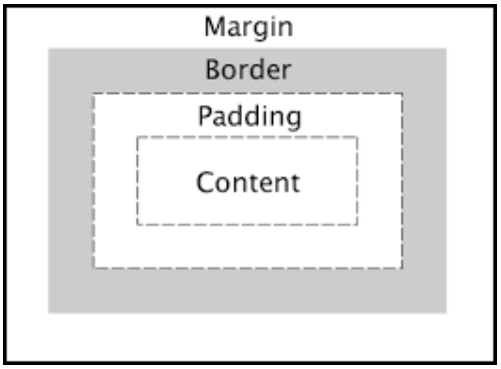
In CSS, the term "box model" is used when talking about design and layout. The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.

Content : The content of the box, where text and images appear.

Padding : Clears an area around the content. The padding is transparent.

Border : A border that goes around the padding and content

Margin : Clears an area outside the border. The margin is transparent



Java Script:

Introduction:

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities. JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name LiveScript. The general-purpose core

of the language has been embedded in Netscape, Internet Explorer, and other web browsers. Javascript helps you create really beautiful and crazy fast websites. The user can develop website with a console like look and feel and give your users the best Graphical User Experience. JavaScript usage has now extended to mobile app development, desktop app development, and game development. This opens many opportunities for you as Javascript Programmer. Due to high demand, there is tons of job growth and high pay for those who know JavaScript. You can navigate over to different job sites to see what having JavaScript skills looks like in the job market. Great thing about Javascript is that you will find tons of frameworks and Libraries already developed which can be used directly in your software development to reduce your time to market.

Client Side JavaScript:

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser. It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

Basic Java Script:

```
<html>
  <body>
    <script language = "javascript" type = "text/javascript">
      <!--
        document.write("Basic print statement")
      //-->
    </script>
  </body>
</html>
```

Bootstrap:

Introduction:

Bootstrap is a sleek, intuitive, and powerful, mobile first front-end framework for faster and easier web development. It uses HTML, CSS and Javascript. It is a free and open-source project,

hosted on GitHub, and originally created by (and for) Twitter. After its open-source release in 2011, Bootstrap became popular very quickly, and not without reason. Web designers and web developers like Bootstrap because it is flexible and easy to work with. Its main advantages are that it is responsive by design, it maintains wide browser compatibility, it offers consistent design by using re-usable components, and it is very easy to use and quick to learn. It offers rich extensibility using JavaScript, coming with built-in support for jQuery plugins and a programmatic JavaScript API. Bootstrap can be used with any IDE or editor, and any server-side technology and language, from ASP.NET to PHP to Ruby on Rails. With Bootstrap, web developers can concentrate on the development work, without worrying about design, and get a good-looking website up and running quickly. Conversely, it gives web designers a solid foundation for creating interesting Bootstrap themes. Bootstrap is available in two forms; as a precompiled version, and as a source code version. The source code version uses the Less CSS pre-processor, but if you are more into Sass, there is an official Sass port of Bootstrap also available. To make it easier to make use of CSS vendor prefixes, Bootstrap uses Auto prefixer. The source code version comes styles source code written in Less (or Sass), all the JavaScript, and accompanying documentation. This allows more ambitious designers and developers to change and customize, at their will, all the provided styles, and to build their own version of Bootstrap. All the styles can be overridden later by using custom styles.

File Structure of Bootstrap:

```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   └── bootstrap-theme.min.css
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings-regular.eot
    ├── glyphicons-halflings-regular.svg
    ├── glyphicons-halflings-regular.ttf
    ├── glyphicons-halflings-regular.woff
    └── glyphicons-halflings-regular.woff2
```

CHAPTER IV SYSTEM IMPLEMENTATION

4.1 TOOLS AND TECHNOLOGIES USED

Python:

Introduction:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP. Interact with the interpreter directly to write your programs. Python supports Object-Oriented style or technique of programming that encapsulates code within objects. Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python:

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Features of python:

Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
Easy-to-read: Python code is more clearly defined and visible to the eyes.
Easy-to-maintain: Python's source code is fairly easy-to-maintain.
A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and portedto many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Installation of Python:

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for platform and install Python. If the binary code in platform is not available, then C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that require in installation.

STEP 1: Select the version of python to install

Python has various versions available with differences between the syntax and working of different versions of the language. We need to choose the version which we want to use or need. There are different versions of Python 2 and Python 3 available.

STEP 2: Download Python Executable Installer

On the web browser, in the official site of python (www.python.org), move to the Download for Windows section. All the available versions of Python will be listed. Select the version required by you and click on Download. Let suppose, we chose the Python 3.9.1 version. On clicking download, various available executable installers shall be visible with different operating system specifications. Choose the installer which suits your system operating system and download the installer. Let suppose, we select the Windows installer (64 bits).

STEP 3: Run Executable Installer

We downloaded the Python 3.9.1 Windows 64-bit installer.

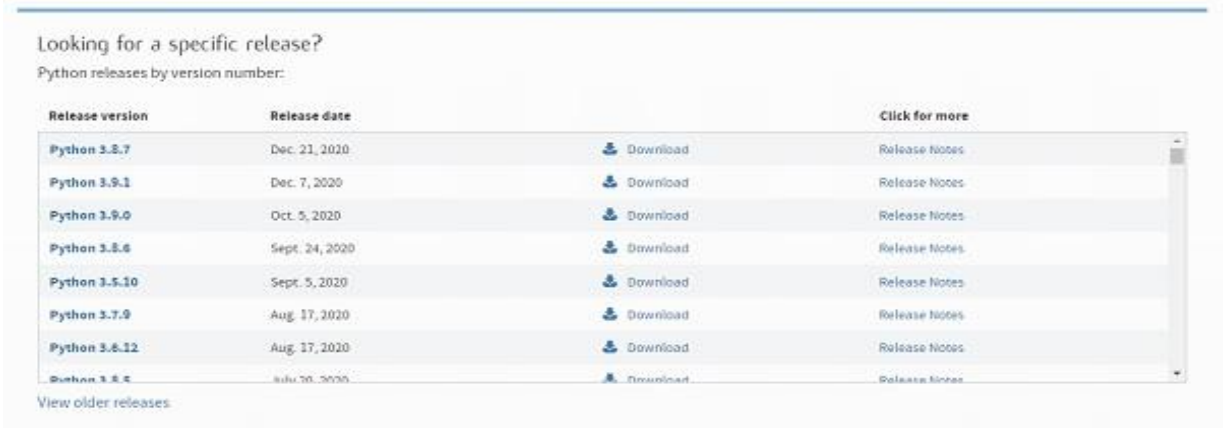
Run the installer. Make sure to select both the checkboxes at the bottom and then click Install New. On clicking the Install Now, the installation process starts. The installation process will take few minutes to complete and once the installation is successful, the following screen is displayed.

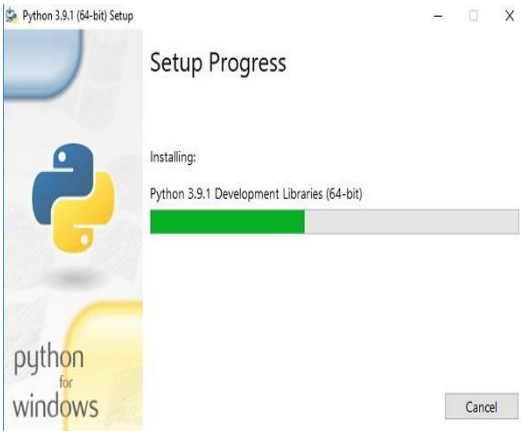
STEP 4: Verify Python is installed on Windows

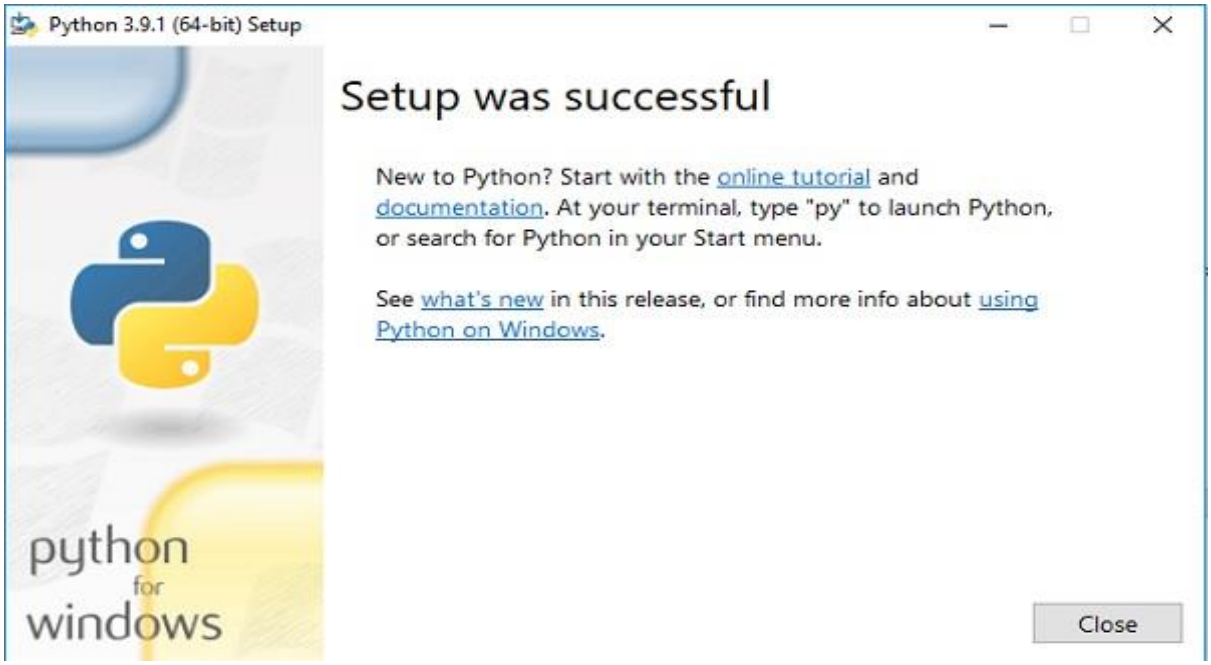
To ensure if Python is successfully installed on your system, follow the given steps: Open the command prompt, Type 'python' and press enter, The version of the python which you have installed will be displayed if the python is successfully installed on your windows.

STEP 5: Verify Pip was installed

Pip is a powerful package management system for Python software packages. Thus, sure, that you have it installed. To verify if pip was installed, follow the given steps Open the command prompt, enter pip –V to check if pip was installed, output appears if pip is installed successfully.







Pip tool usage:

pip is the package installer for Python. You can use it to install packages from the Python Package Index and other indexes. Pip tools are a set of third-party command line tools designed to help with Python 3 dependency management by keeping dependencies up to date, even when you have them pinned. You can install it by running the following command: python -m pip install pip-tools. Pip alone is sufficient to install from pre-built binary archives, up to date copies of the setup tools and wheel projects are useful to ensure you can also install from source archives.

Installation command: python -m pip install pip-tools

XAMPP:

Introduction:

XAMPP is one of the widely used cross-platform web servers, which helps developers to create and test their programs on a local webserver. It consists of Apache HTTP Server, MariaDB, and interpreter for the different programming languages. It is available in 11 languages and supported by different platforms such as the IA-32 package of Windows & x64 package of macOS and Linux. XAMPP is an abbreviation where *X stands for Cross-Platform, A stands for Apache, M stands for MySQL, and the P's stand for PHP and Perl,* respectively.

XAMPP Format Support:

XAMPP is supported in three file formats, .exe is an extension used to denote executable files making it accessible to install because an executable file can run on a computer as any normal program., the .exe is the most straightforward format to install, while the other two formats are quite complicated and complex to install., .7z - 7zip file extension is used to denote compressed files that support multiple data compression and encryption algorithms. It is more favoured by a formalist, although it requires working with more complex files., .zip extension supports lossless compression of files, a zipped file may contain multiple compressed files. The Deflate algorithm is mainly used for compression of files supported by this format. The .ZIP files are quite tricky to install as compared to .exe.

XAMPP Controller:

XAMPP Control Panel is a management tool that offers to supervise the actions of individual components of XAMPP. It controls each component of the text server. The user can initiate or halt discrete modules by operating upon the buttons like "Actions" column. Control panel efficiently manage all the components of the XAMPP.

Operating XAMPP Control Panel:

STEP 1: Open the XAMPP Control Panel by clicking on the icon. If the icon is not visible then, go to, All Programs → Apache Friends → XAMPP → XAMPP Control Panel.

STEP 2: Click Start button corresponding to Apache and MySQL. Establishing and terminating services physically is considered better when one does not include the Apache and MySQL components to run for a considerable time.

STEP 3: Initially, while starting Apache or MySQL, Windows security will question to Allow access to the servers on your local network in order to unblock the servers and grant access to the system.

STEP 4: The "Actions" button is a toggle button that switches between Start/Stop. The Port information is also available corresponding to the modules once they are active.

STEP 5: Click the Close × button in at the upper right end, which will enable XAMPP to run in the background. It will now be accessible through the Notification area.

STEP 6: To manage the XAMPP Control Panel while running in the background, you can simply right-click the XAMPP icon in the notification area and start/stop components without opening it.

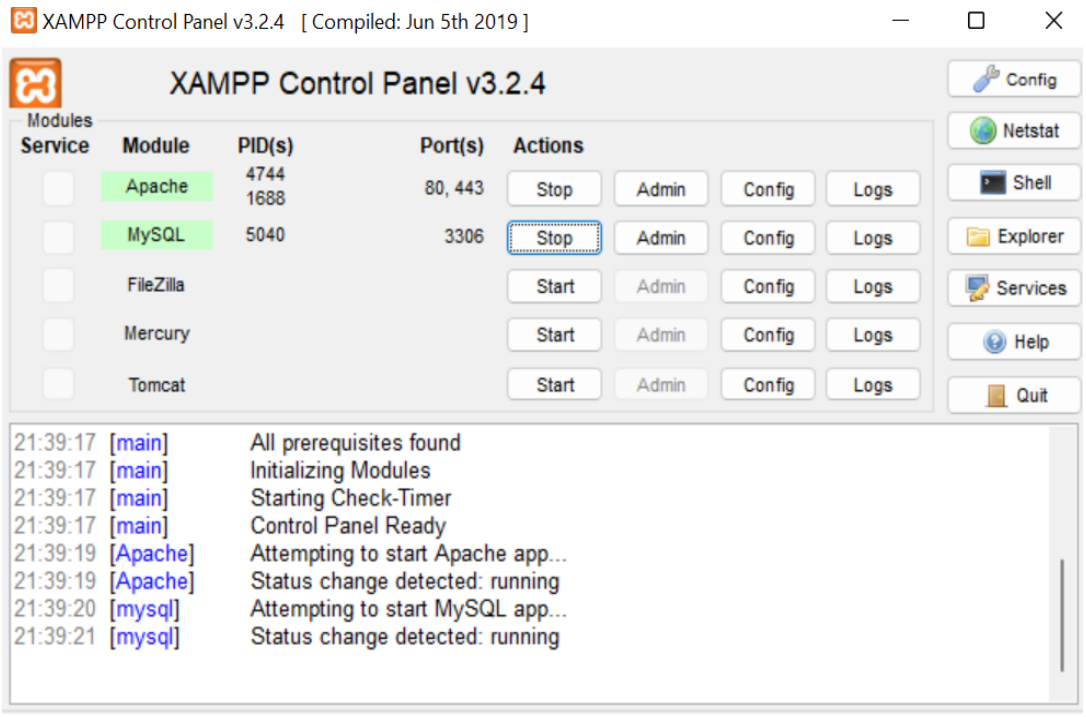
STEP 7: Admin and Config buttons for Modules:

Admin: In the case of Apache, it opens XAMPP for Windows Admin page in the default browser, and for MySQL, it redirects to PhpMyAdmin in the default browser.

Config: In the case of Apache, it provides access to Apache folders and configuration files, like httpd.conf, and for MySQL, it grants access to MySQL database folders.

STEP 8: In order to suspend the running of any of the components like Apache or MySQL, click the "Stop" button corresponding to the module you wish to stop.

STEP 9: To stop the Control Panel from running as a background application, click on the the "Quit" button at the lower right end. This will remove the XAMPP Launcher from the Notification Bar. To operate again, XAMPP Control Panel needs to be Re-launched.



MySQL:

Introduction:

MYSQL is one of the most popular database management systems (DBMSs) on the market today. As most software applications need to interact with data in some form, programming languages like Python provide tools for storing and accessing these data sources. It will develop a small MySQL database and learn how to query it directly from your Python code. MySQL is also a part of the Oracle ecosystem. While its core functionality is completely free.

Steps to create MySql databse using XAMPP:

STEP 1: Navigate to XAMPP in your system or simply launch it by clicking the XAMPP Icon. The Control Panel is now visible and can be used to initiate or halt the working of any module.

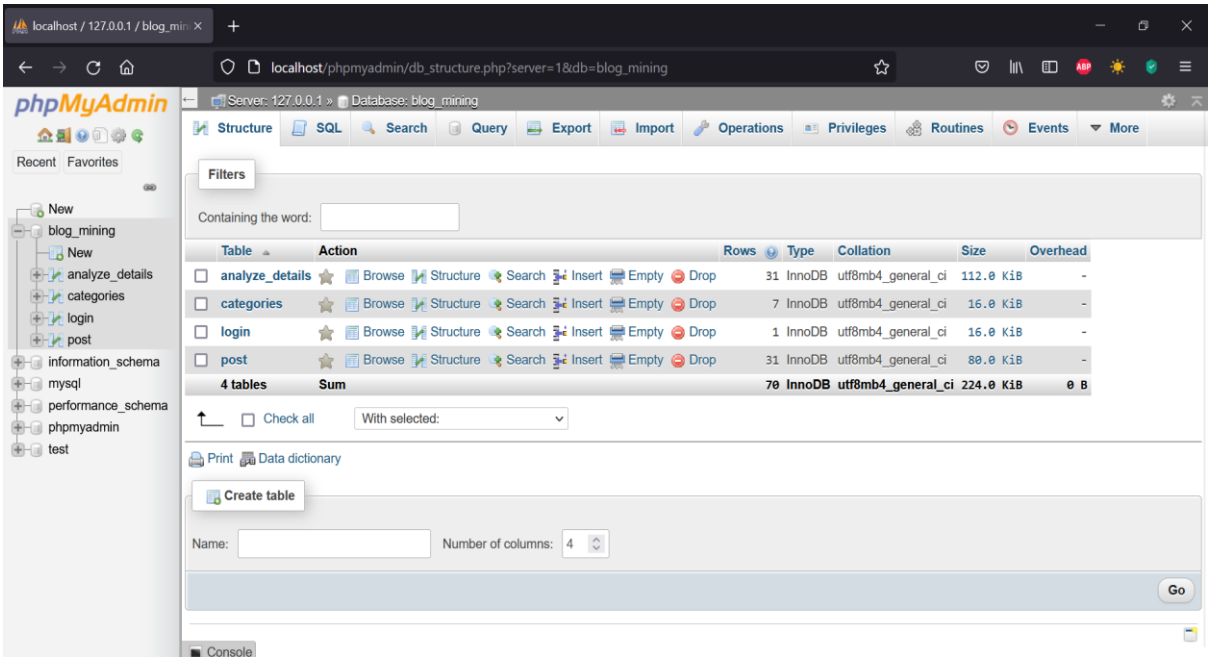
STEP 2: Click on the "Start" button corresponding to Apache and MySQL modules. Once it starts working, the user can see the screen

STEP 3: Now click on the "Admin" button corresponding to the MySQL module. This automatically redirects the user to a web browser to the following address- http://localhost/phpmyadmin

STEP 4: One can see a number of tabs such as Database, SQL, User Accounts, Export, Import, settings, etc. Click on the "Database" tab. Here you can see the Create option. Choose an appropriate name for the input field titled Database name. Things to keep in mind while selecting the name for the database are - The number of characters used should be equal to or less than 64. The name should comprise of letters, numbers and under score. The DB name should not start with a number. It should be relevant to the topic for which it is being created. Make sure the database is successfully created.

STEP 5: It is very important to create tables in order to store the information in a systematic manner. In this step, we will build tables for the created database. In the created Database (Login page in this case), click on the 'Structure' tab. Towards the end of the user will see a 'Create Table' option. Fill the input fields titled "Name" and “Number of Columns” and hit the ‘Go’ button.

STEP 6: Now, we have to initialize our columns based on their type. Enter the names for each of your columns, select the type, and the maximum length allowed for the input field. Click on "Save" in the bottom right corner. The table with the initialized columns has been created. You can create any number of tables for your database.



Flask app:

Introduction:

Flask is a small web application framework. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects. Flask is classified as a micro-framework because it does not necessitate the use of any specific tools or libraries. Side tabs are used to display different contents or for single-page web applications.

Installation of Flask MySql:

Flask connects to MySQL via the flask mysqldb connector. To install the package, use the command “pip install flask_mysqldb”. Setting Up Flask MySQL Database is as follows:

STEP 1: Connecting a Flask Application to a MySQL Database: To connect application import Flask, render_template, request package from flask module and MySQL from flask_mysqldb module. then configure host, user, password, flask by below commands:

```
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'flask'
mysql = MySQL(app)
```

STEP 2: Configuring the MySQL Connection Cursor: Cursor thus provides a means for Flask to interact with the database tables. It can scan the database for data, execute SQL queries, and delete table records. Create a connection Cursor by using: "cursor = mysql.connection.cursor()"

STEP 3: Programming a Flask Application: the Flask application that will store user submitted data in the MySQL DB Table. When the user submits the data, the cursor inserts it into the MySQL DB.

STEP 4: Putting the Code into Action: Enter the information and press the Submit button. Took a look at it in the phpMyAdmin web interface now.

Running flask app:

STEP 1: Set up Working Directory.

STEP 2: Setup the Virtual Environment.

STEP 3: Create the Virtual Environment.

STEP 4: Activate the Virtual Environment.

STEP 5: Upgrade pip.

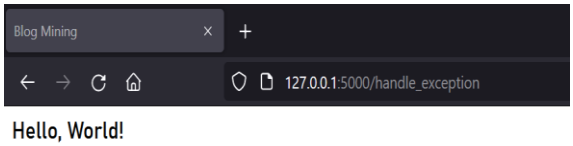
STEP 6: Install Flask.

STEP 7: Make available dependencies to a file.

STEP 8: On terminal, use command python file_name.py

STEP 9: Copy the URL, and paste it on browser

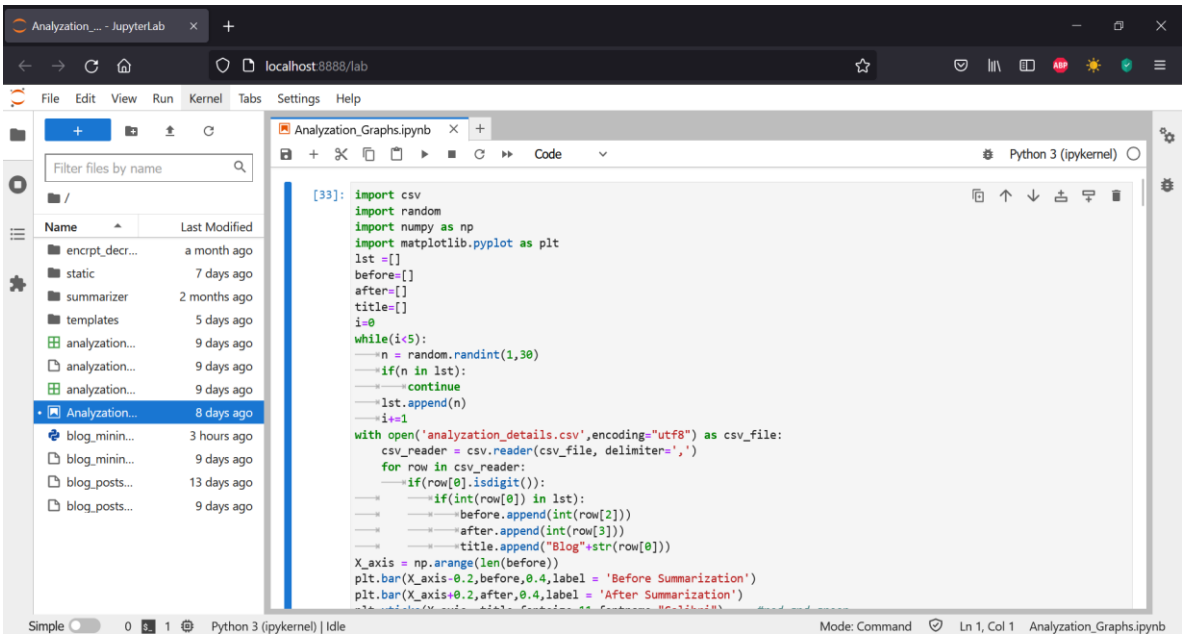
```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"

if __name__=="__main__":
    app.run()
```



Jupyter Notebook:

The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser. The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access. It is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Its uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. The IPython Notebook is now known as the Jupyter Notebook. It is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media.

STEP 1: To Install jupyterLab with pip use “pip install jupyterlab” in command prompt.
STEP 2: Now install classic jupyter Notebook with “pip install notebook”.
STEP 3: Once installed, the command “jupyter-lab” is used to launch it.
STEP 4: Run the notebook by using “jupyter notebook”.



Sublime Editor Tool:

Introduction:

Sublime Text is a sophisticated text *editor* for code, markup and prose. Sublime Text is a shareware cross-platform source code editor. It natively supports many programming languages and markup languages. Users can expand its functionality with plugins, typically community-

built and maintained under free-software licenses. To facilitate plugins, Sublime Text features a Python API. It is such sophisticated text editor which is widely used among developers. It includes wide features such as Syntax Highlight, Auto Indentation, File Type Recognition, Sidebar, Macros, Plug-in and Packages that make it easy for working with code base. It is an Integrated Development Editor (IDE). The benefits of sublime are: Ability to solve linker errors, keeping track of all files and folders to work with, Connectivity with version control systems like Git, Mercurial, problem-solving capabilities, keeping colour combination for syntax combination.

History of sublime:

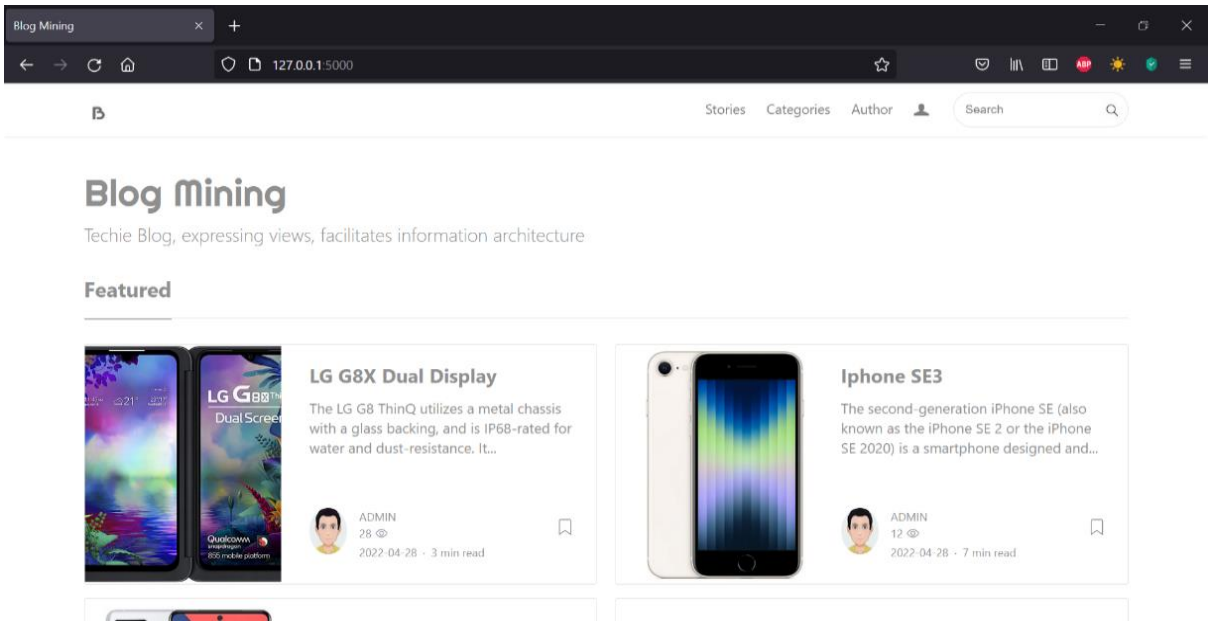
Jon Skinner left his job as a software engineer at Google in order to pursue a dream: to build a better text editor. The end result is the popular Sublime Text today. The initial version of Sublime Text was released to the public on January 18, 2008. Its GUI is similar to the version we are using today.

Features:

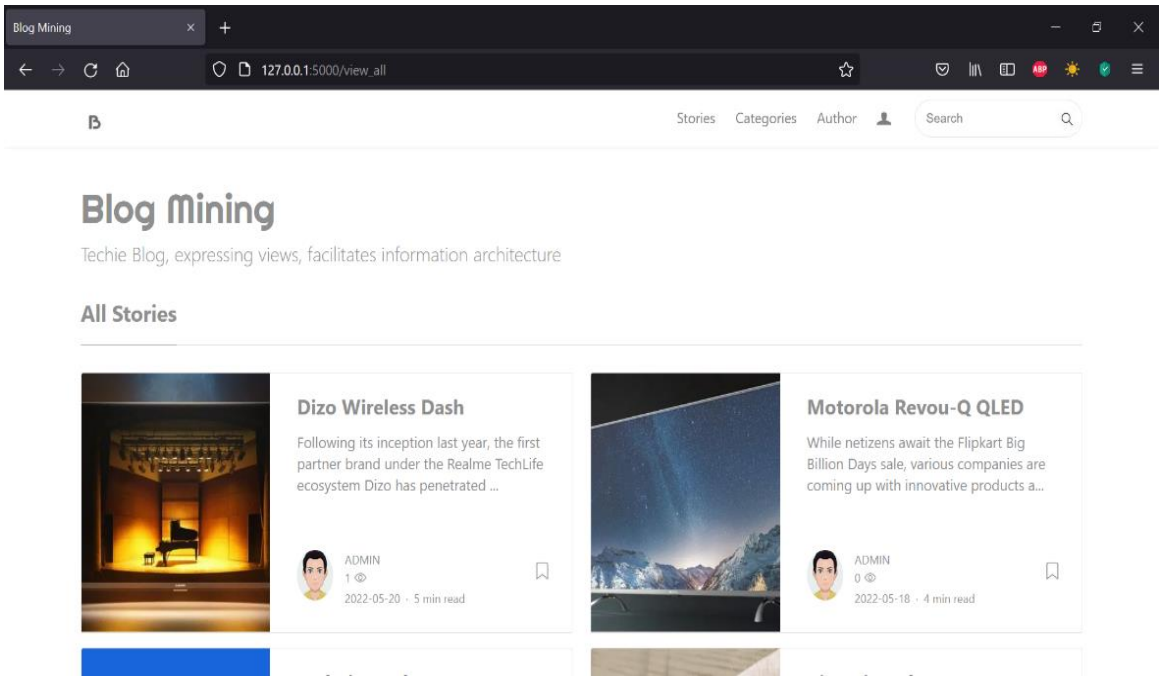
1. Syntax Highlight
2. Auto Indentation
3. File type recognition
4. Side bar with files of mentioned directory
5. Macros
6. Plug-in and Packages

4.2 SCREENSHOTS

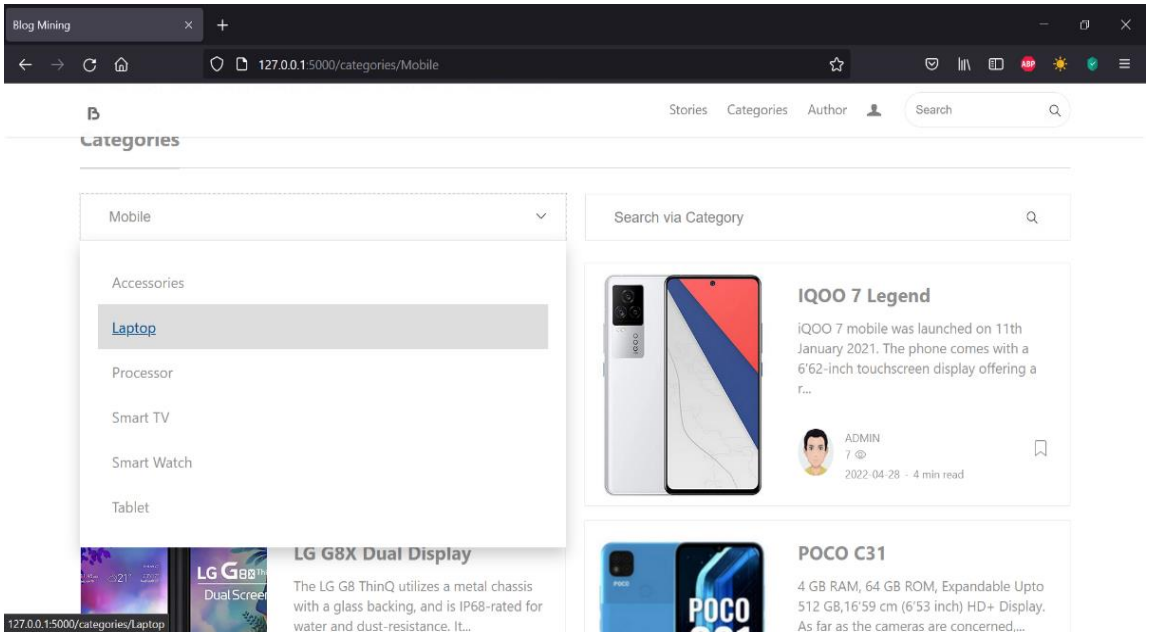
Blog Home



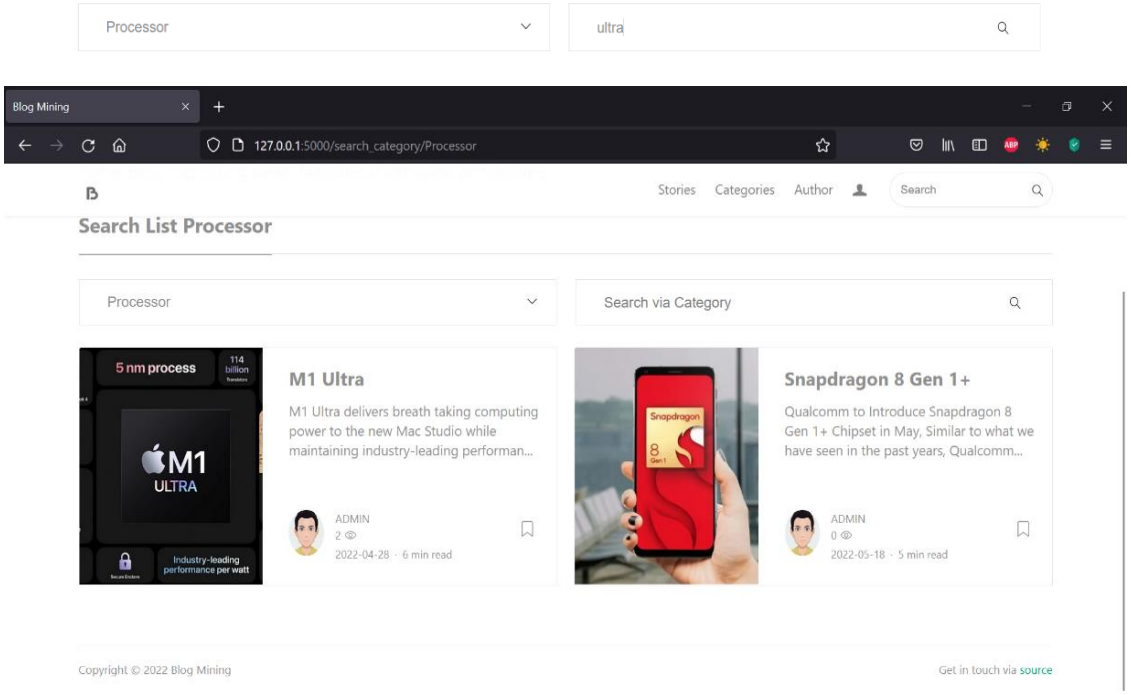
View All



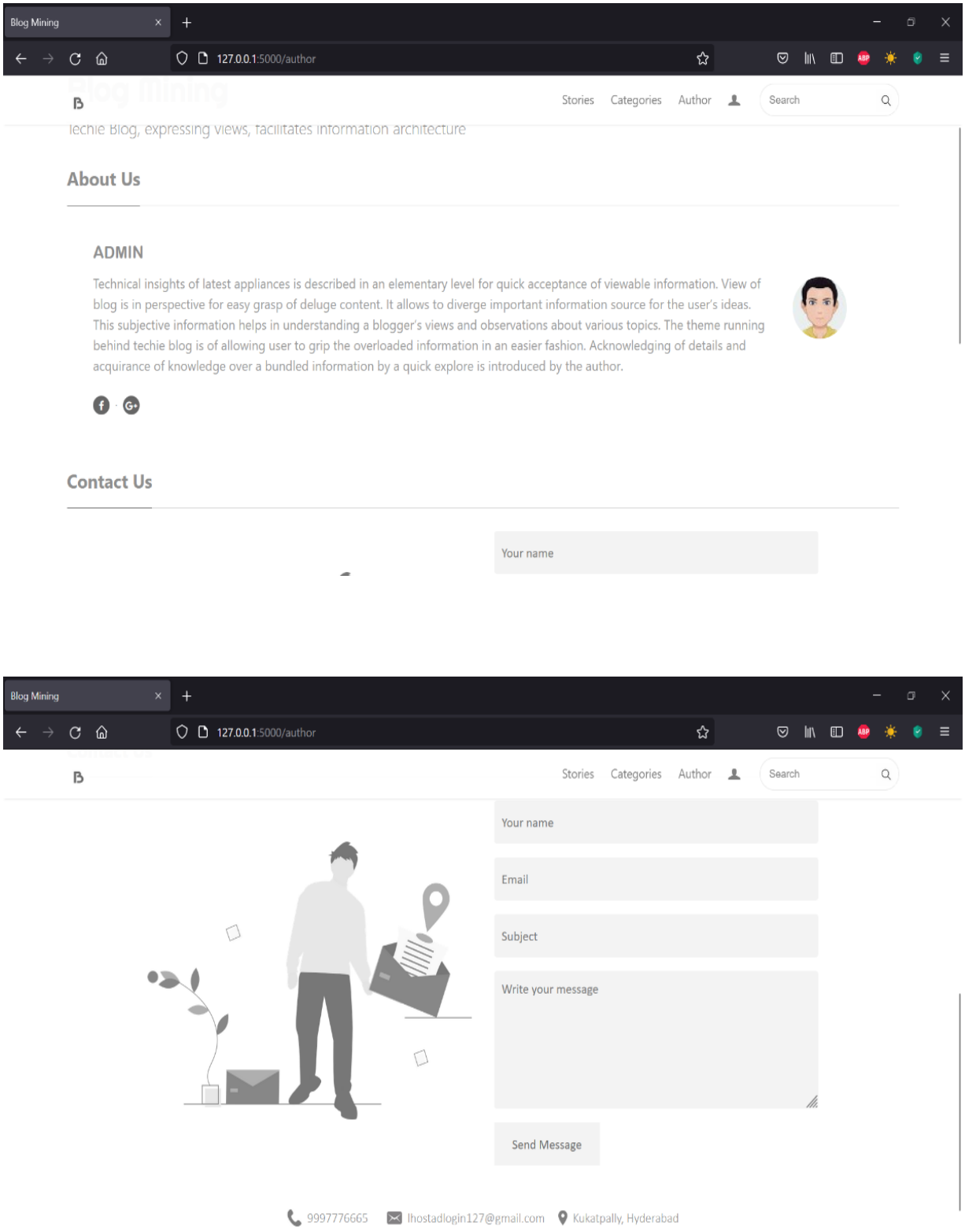
Categories



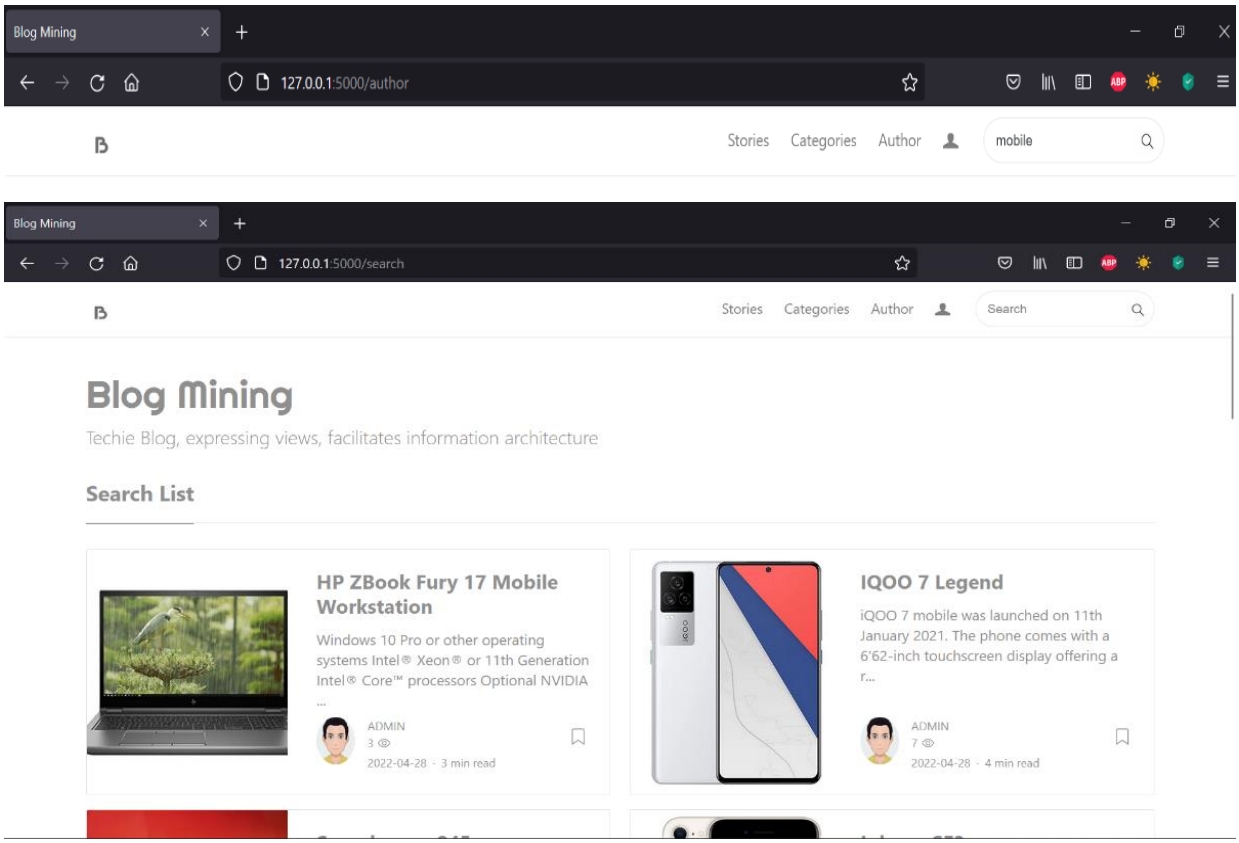
Search via Categories



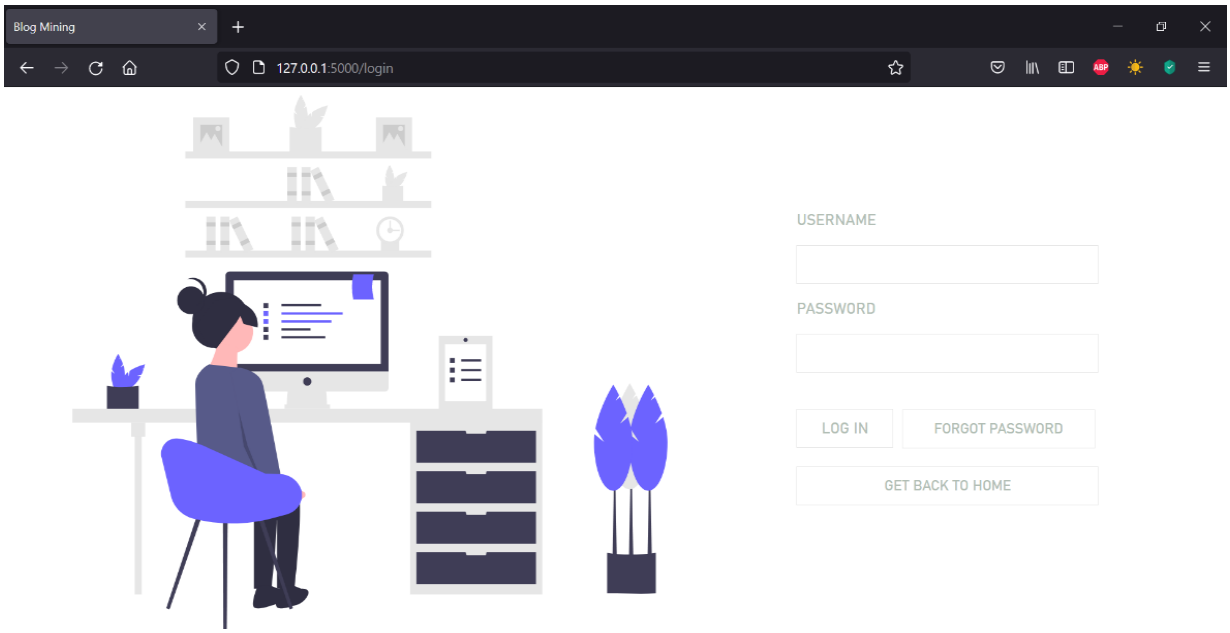
Author


Blog Mining
127.0.0.1:5000/author
Stories
Categories
Author
Search
About Us
ADMIN
Technical insights of latest appliances is described in an elementary level for quick acceptance of viewable information. View of blog is in perspective for easy grasp of deluge content. It allows to diverge important information source for the user's ideas. This subjective information helps in understanding a blogger's views and observations about various topics. The theme running behind techie blog is of allowing user to grip the overloaded information in an easier fashion. Acknowledging of details and acquirance of knowledge over a bundled information by a quick explore is introduced by the author.
Contact Us
Your name
Email
Subject
Write your message
Send Message
9997776665
lhostadlogin127@gmail.com
Kukatpally, Hyderabad

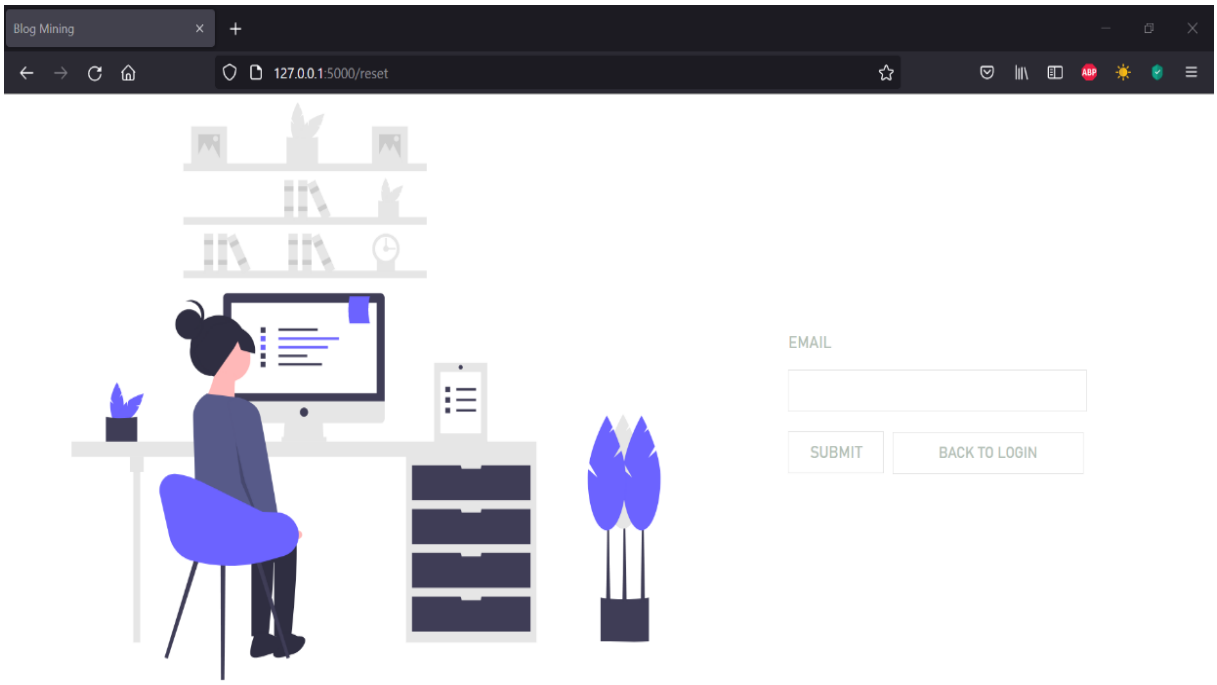
Search via blogs

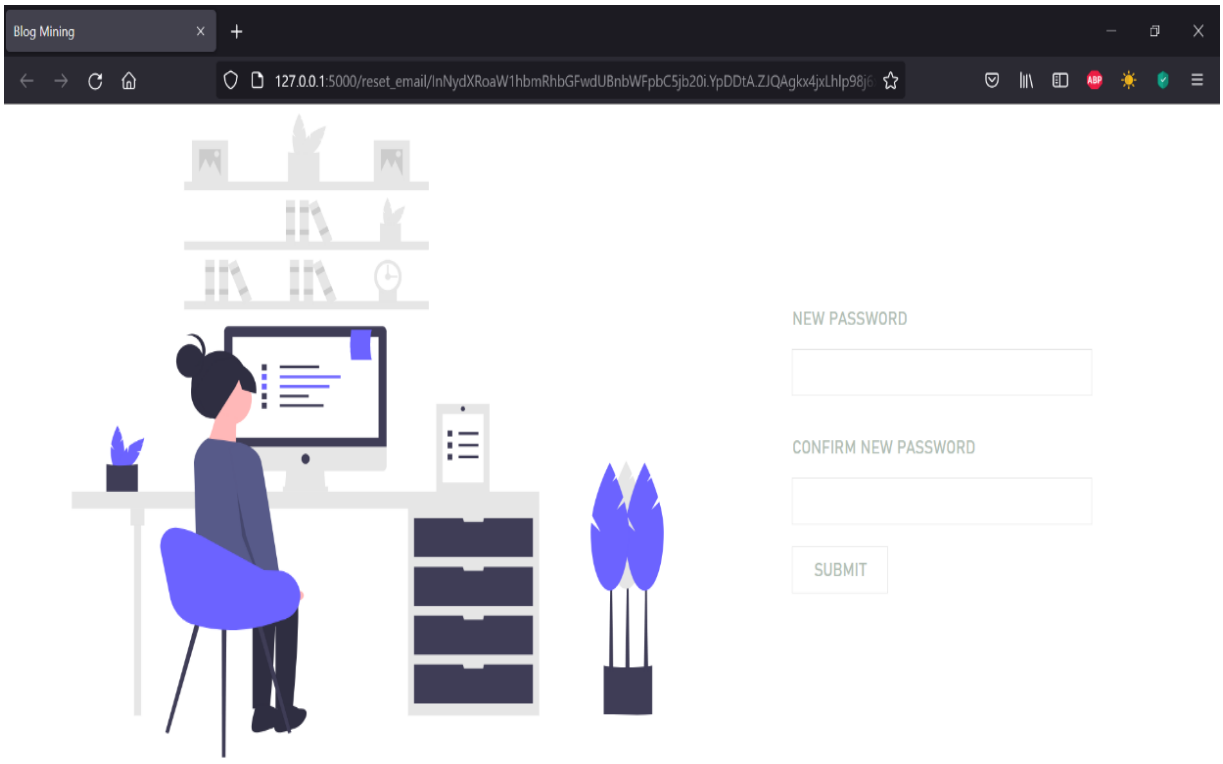


Admin Login

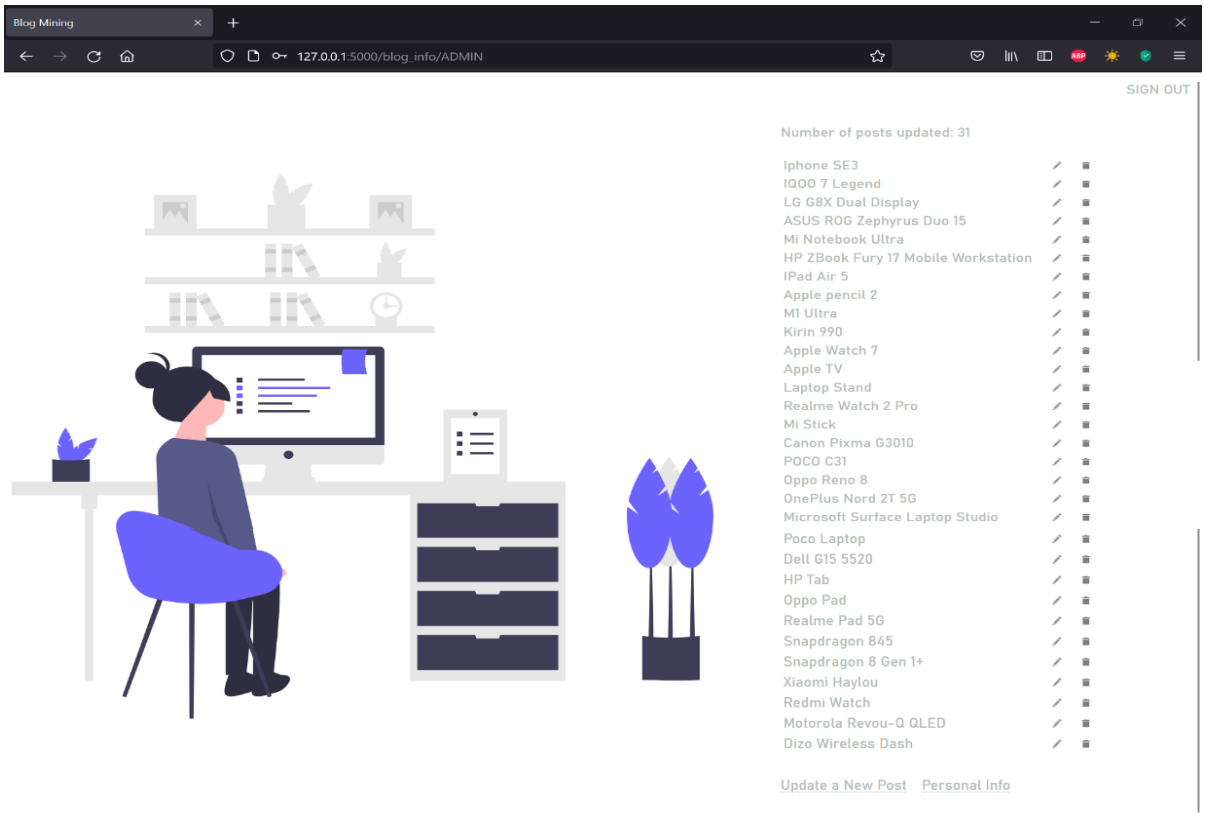


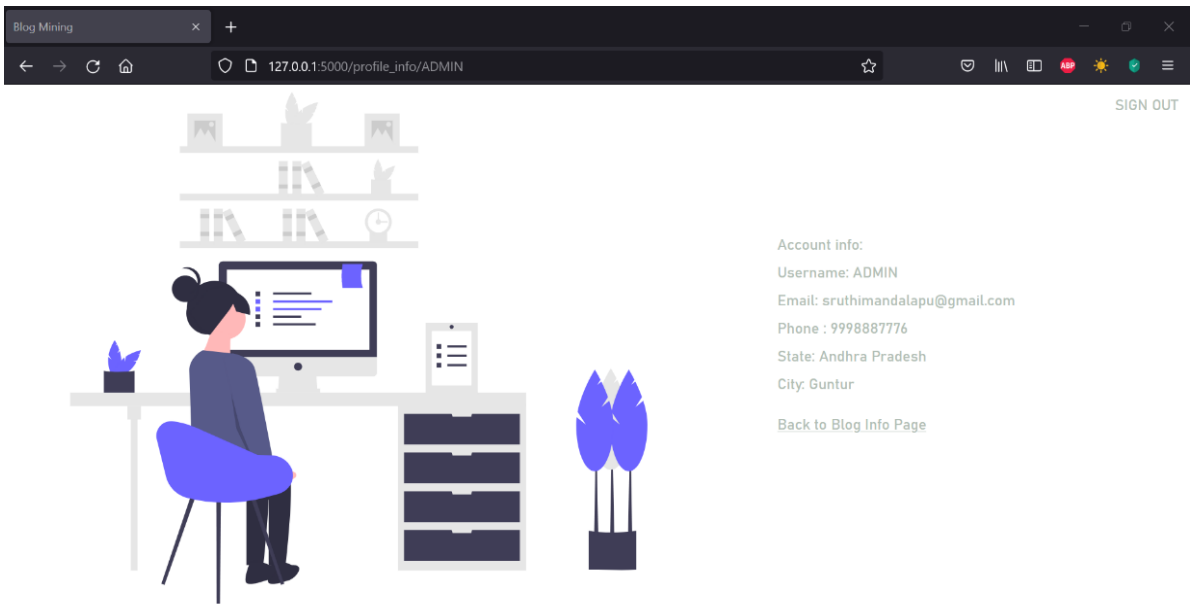
Admin Reset


Blog Mining
127.0.0.1:5000/reset
EMAIL
SUBMIT
BACK TO LOGIN

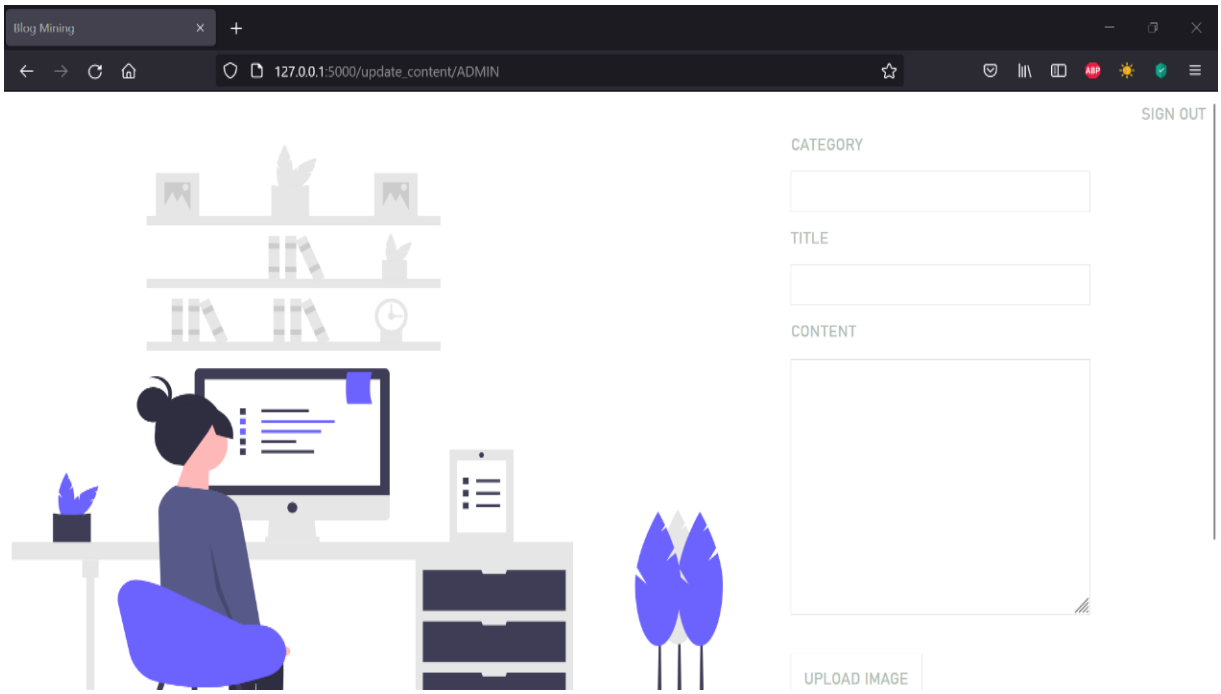

Blog Mining
127.0.0.1:5000/reset_email/InNydXRoaW1hbmRhbGFwdUBnbWFpbC5jb20i.YpDDtA.ZJQAgkx4jxLhIp98j6
NEW PASSWORD
CONFIRM NEW PASSWORD
SUBMIT

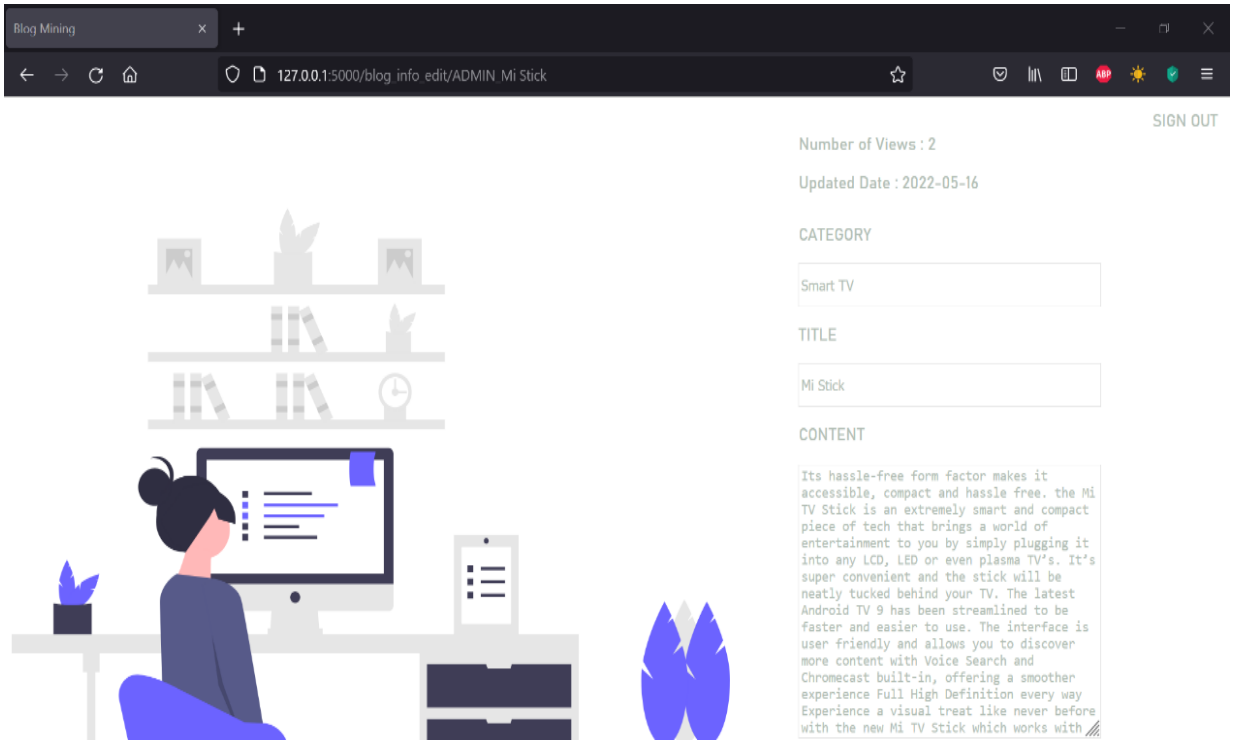
Blog_info, Profile_info



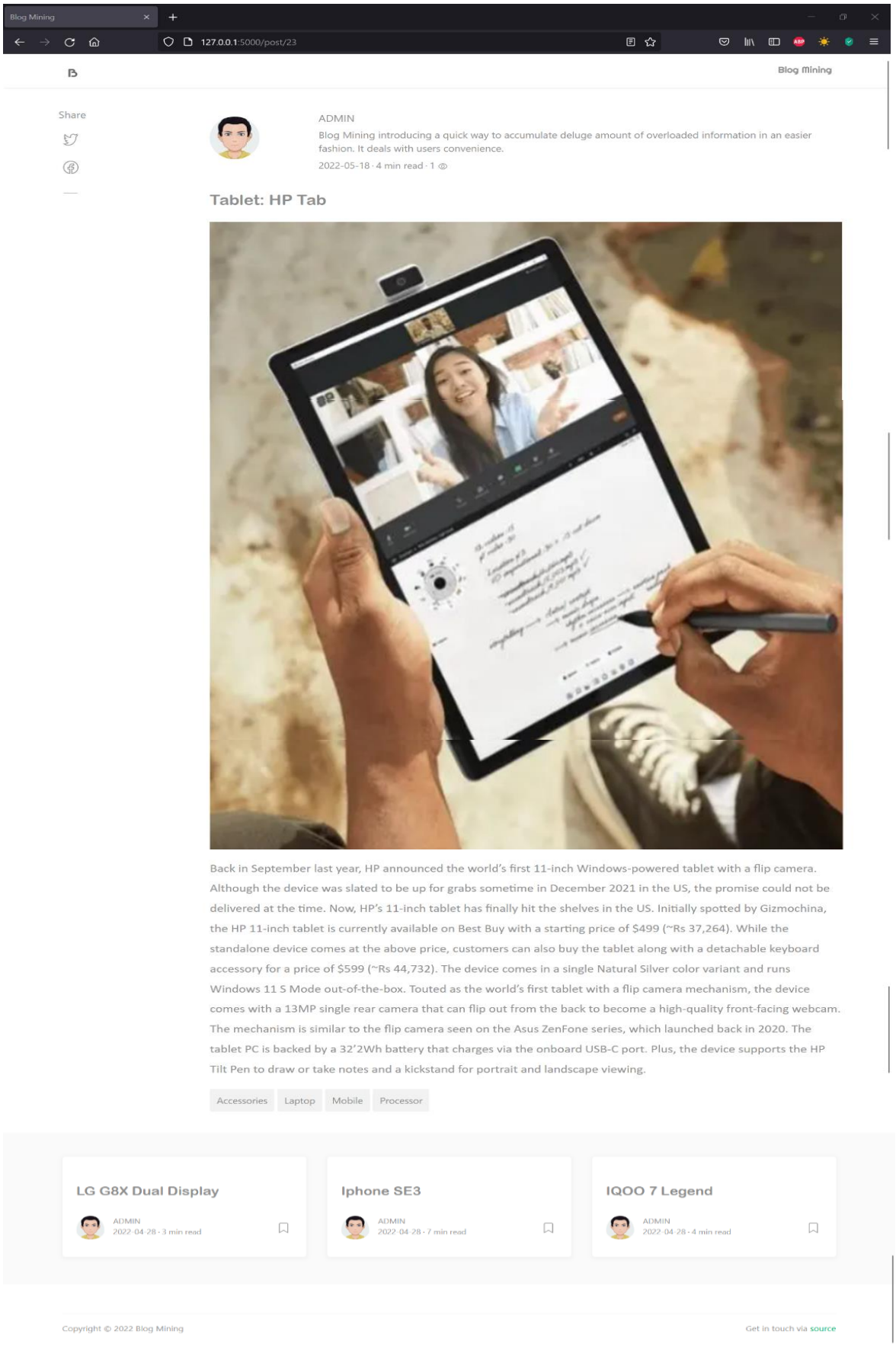


Updation of new blog, edit any of blog updated

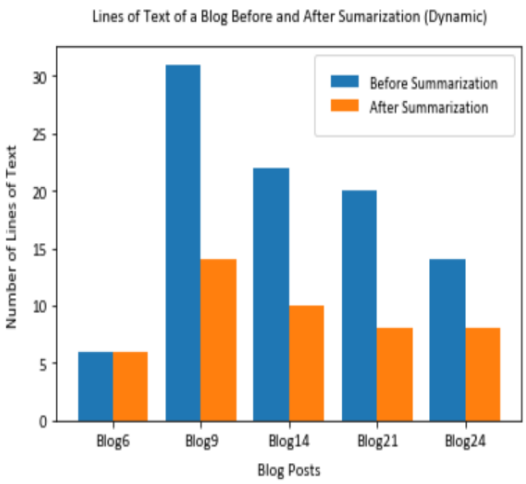


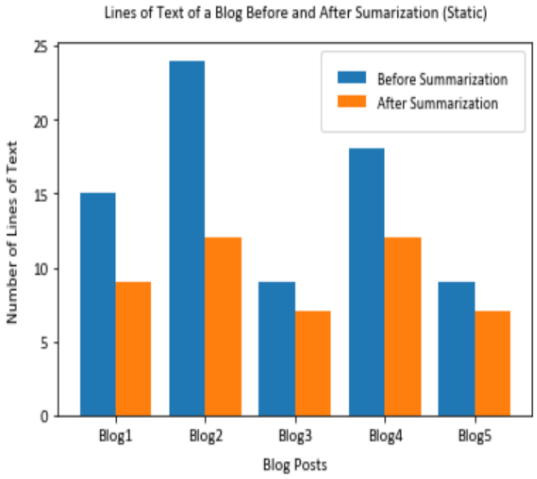


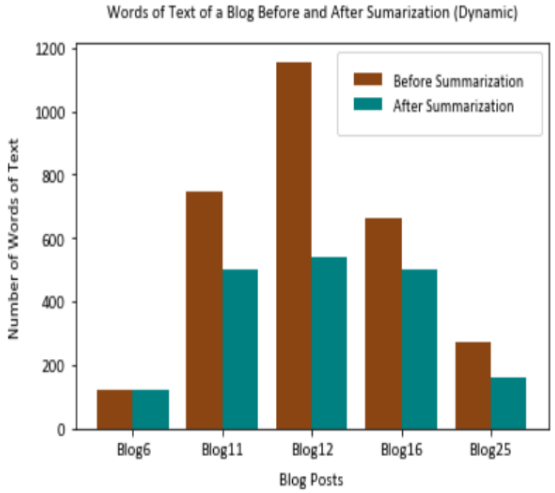
Visualize respective post contents

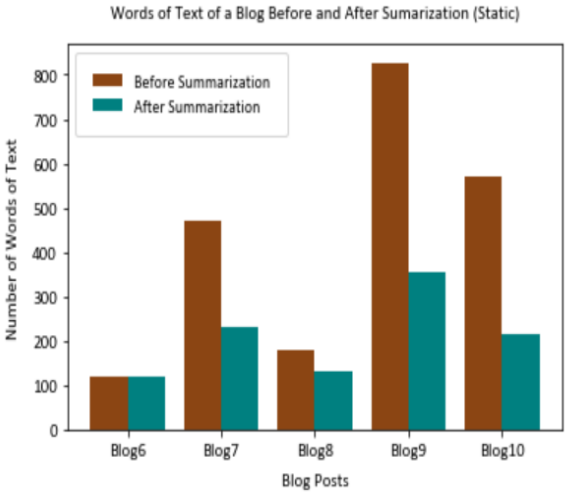


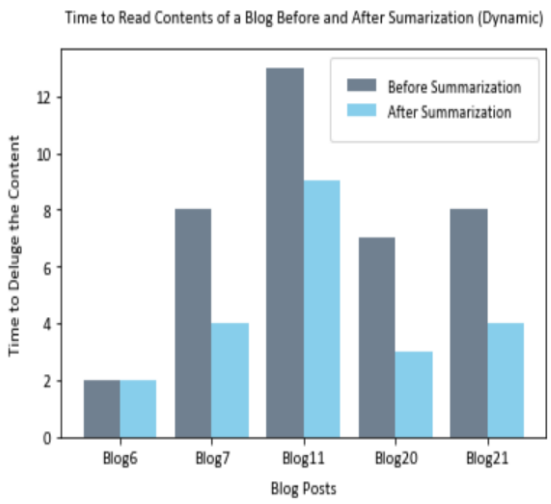
Analysis (Before Summarization and After Summarization)

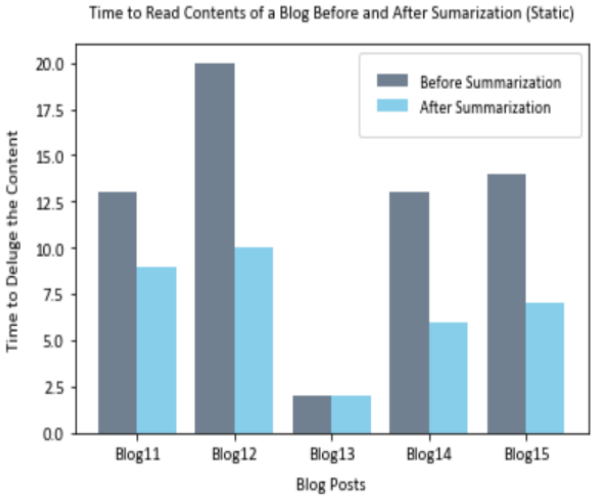


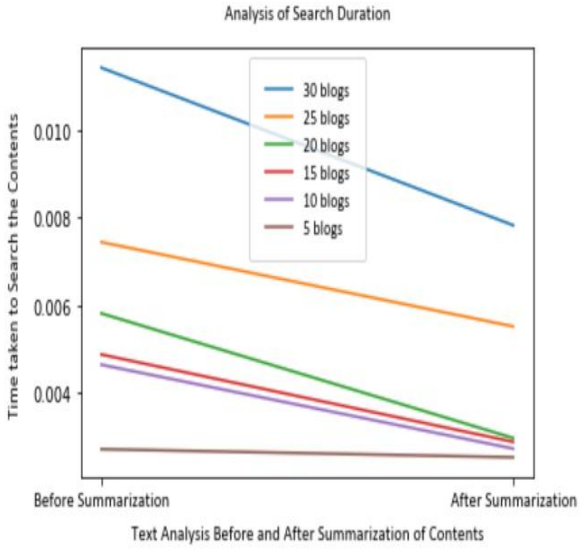


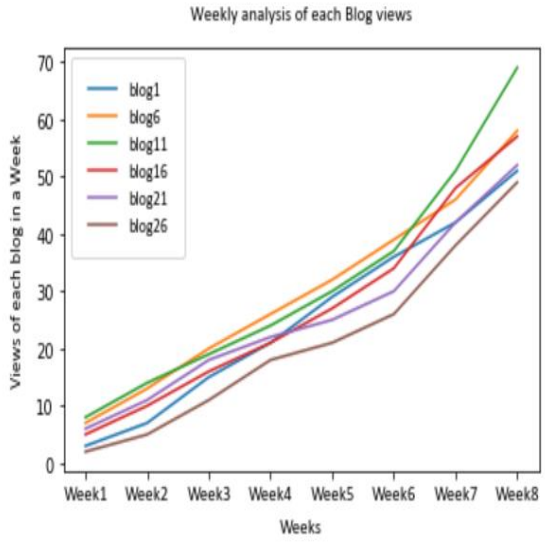





Time to Read Contents of a Blog Before and After Sumarization (Dynamic)
Time to Deluge the Content
Before Summarization
After Summarization
0
2
4
6
8
10
12
Blog6
Blog7
Blog11
Blog20
Blog21
Blog Posts


Time to Read Contents of a Blog Before and After Sumarization (Static)
Time to Deluge the Content
Before Summarization
After Summarization
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Blog11
Blog12
Blog13
Blog14
Blog15
Blog Posts


Analysis of Search Duration
Time taken to Search the Contents
30 blogs
25 blogs
20 blogs
15 blogs
10 blogs
5 blogs
0.004
0.006
0.008
0.010
Before Summarization
After Summarization
Text Analysis Before and After Summarization of Contents


Weekly analysis of each Blog views
Views of each blog in a Week
blog1
blog6
blog11
blog16
blog21
blog26
0
10
20
30
40
50
60
70
Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8
Weeks

CHAPTER V SYSTEM TESTING

5.1 INTRODUCTION

System testing is a process of testing the entire system that is fully functional, in order to ensure the system is bound to all the requirements provided by the client in the form of the functional specification or system specification documentation. In most cases, it is done next to the Integration testing, as this testing should be covering the end-to-end system's actual routine. This type of testing requires a dedicated Test Plan and other test documentation derived from the system specification document that should cover both software and hardware requirements. By this test, we uncover the errors. It ensures that all the system works as expected. We check System performance and functionality to get a quality product. System testing is nothing but testing the system as a whole. This testing checks complete end-to-end scenarios as per the customer's point of view. Functional and Non-Functional tests also done by System testing. All things are done to maintain trust within the development that the system is defect-free and bug-free. System testing is also intended to test hardware/software requirements specifications. System testing is more of a limited type of testing; it seeks to detect both defects within the "inter-assemblages".

5.1.1 SYSTEM OVERVIEW

The typical blogging system consisting of front end, backend interacting with mysql. Each page is further linked with other pages need to be tested. The algo latent semantic analysis, which was running summarizes the contents and updates to the database. The updation of blog content can only be done by admin. Admin logs in and edit, delete, insert a new blog based on necessary requirements. The updated here there by visualized in home page, and can also visualize all stories. The typical blog content is a summarized content which is easy for digest and even search puts quicker.

5.1.2 TEST APPROACH

Black box Testing:

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer. This testing also is

known as behavioural testing. Black-box testing mainly focuses on input and output as the internal code is hidden from the tester. In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.

Functional Testing:

Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements. Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user's expectations. This testing makes sure that the functionality of a product is working as per the requirements specification, within the capabilities of the system. Functional testing is done manually or with automated tools.

Unit Testing:

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff. It is testing the smallest testable unit of an application. It is done during the coding phase by the developers. To perform unit testing, a developer writes a piece of code (unit tests) to verify the code to be tested (unit) is correct.

Error Handling Testing:

To determine the ability of the System to process erroneous transactions properly. All reasonable error is supposed to detect by the application system. Control over the error during error correction is a must. Procedures mostly guarantee that errors are get corrected properly. This testing should happen throughout SDLC. Errors encompass all unexpected conditions. It checks the ability of the software to execute all transactions properly.

Integration Testing:

Integration testing -- also known as integration and testing (I&T) -- is a type of software testing in which the different units, modules or components of a software application are tested as a combined entity. However, these modules may be coded by different programmers. Integration testing ensures that an entire, integrated system meets a set of requirements. It is performed in an integrated hardware and software environment to ensure that the entire system functions properly.

Performance Testing:

This testing makes sure the system's performance under various conditions, in terms of performance characteristics. This testing is also called compliance testing with respect to performance. It checks when multiple users use the same app at a time, then how it responds back.

Regression Testing:

Each time a new module is added as a part of integration as the software changes. Regression testing is an actually that helps to ensure changes that do not introduce unintended behaviour as additional errors. Regression testing maybe conducted manually by executing a subset of all test cases or using automated capture play back tools enables the software engineer to capture the test case and results for subsequent playback and compression. The regression suit contains different classes of test cases. A representative sample to tests that will exercise all software functions.

5.2 TEST PLAN

A Test Plan refers to a detailed document that catalogs the test strategy, objectives, schedule, estimations, deadlines, and the resources required for completing that particular project. Think of it as a blueprint for running the tests needed to ensure the software is working properly – controlled by test managers.

5.2.1 FEATURES TO BE TESTED

Front end view:

The Blog Mining and Emotion argumentation typically contains Blog home page, categories,

author, search the contents, respective admin login page and reset part, on login having blog info page and edit and updation of blogs and access of profile page. Each page is tested with other alternate page via integration, the divergence of linked urls. Each part of the page verified by checking functionality with the verification of output insights.

Back end view:

It is the verification of database weather the content loaded from mysql database is proper any case of errors is tested. It is verified as a part of unit test, each block of code is verified on apart. Typically considering a function of flask app is testing at once on its working, considering that considerable outcomes had put out.

Algo part:

The algo Latent Semantic Analysis is tested on provision of different inputs, this algo allows to put out summarized content, it verified on giving input the content and number of lines to be summarized, does the outcome leads to same number of lines of summarized content is verified. The verification of summarized content with actual content how much actually summarized is observed. Providing different input putting corner test cases having empty string as input, providing huge content, providing 0 lines to summarize.

5.2.2 FEATURES NOT TO BE TESTED

The front end part which is a static view, may not be tested which is a refer for look and feel via aesthetics. The page design, the template is a static view having no attaching urls may not require any kind of testing. During analysis the outcome is graphical view which doesn't need functionality to be checked apart. The database view is verified which doesn't require any part of evaluation.

5.2.3 TESTING TOOLS AND ENVIRONMENT

A testing environment is a setup of software and hardware for the testing teams to execute test cases. In other words, it supports test execution with hardware, software and network configured. A test environment enables you to create identical environments every time you need to test your product. It's the most important tool for a testing engineer in order to have confidence in the testing results.

Key Elements of creating a Test Environment:

1. Create test data and insert to test environment (test bed)
2. Set up database
3. Configure the environment
4. Select the right hardware and operating system (e.g. evaluate the difference between running application on Windows 8.1 and Windows 10)

Python testing tool:

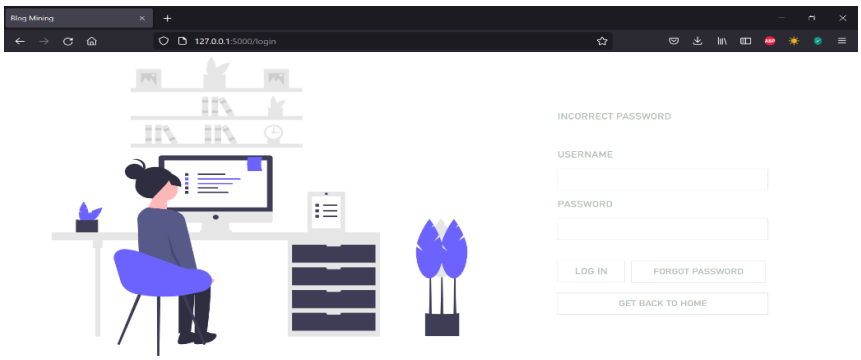
Python provides the unit test module to test the unit of source code. The unit test plays an essential role when we are writing the huge code, and it provides the facility to check whether the output is correct or not. Normally, we print the value and match it with the reference output or check the output manually. This process takes lots of time. To overcome this problem, Python introduces the unit test module. We can also check the application's performance by using it. Python offers a set of tools and libraries which help us to create automated tests for the application. Unit testing is a smaller test, it checks a single component that it is working in right way or not. Using the unit test, we can separate what necessities to be fixed in our system.
Example: assert sum([2, 3, 5]) == 10, "Should be 10"

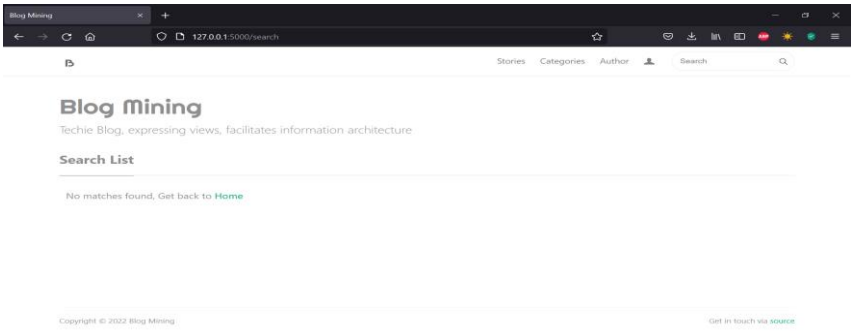
5.3 TEST CASES

A Test Case is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, postcondition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer. A test case is typically a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement. Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

Functional Testcases	Type – Positive/Negative
1. Perform actions on Home page, diverge to categories,	Positive

	Author page, admin login.	
2.	Perform actions on Categories page, diverge to categories, Author page, admin login.	Positive
3.	Perform actions on Author page, diverge to categories, Author page, admin login.	Positive
4.	Visualize post contents from home, categories, view_all Search page, serach via contents page.	Positive
5.	Searching a contents of a blog, view of proper outcome matching with expected result. Providing empty search string, giving unknown search string.	Positive
6.	The same search technique is performed on search via categories.	Positive
7.	Admin Login page, giving proper username and password, giving wrong username and password, giving proper username and wrong password and providing wrong username and correct password.	Positive
8.	Reset page check over new password and confirm new password.	Positive
9.	Verify proper updation and editing of blog contents	Positive
10.	Verify deletion of respective blog content	Positive





CHAPTER VI CONCLUSION

6.1 CONCLUSION

The study of blog mining is used to analyze and search the online blog posts relevant contents in a quite simpler fashion. The main purpose includes provision of summarize content. As the information reflected is summarized view one can grasp away huge content in shorter duration and also browsing gives faster pace results. Blog mining can be concluded as a view by pulling out search optimization and also allows users to acknowledge deluge information of blogging websites.

6.2 FUTURE ENHANCEMENTS

In future we can implement with various languages like Hindi. This allows more enrichment for users who prefer native language, the source having first language allows many users to put interest to look through vivid contents of the blogs.

CHAPTER VII APPENDICES

7.1 SAMPLE CODE SNIPPETS

```
import os
from datetime import date
from flask import Flask,render_template,request,url_for,redirect,flash
from flask_mail import Mail, Message
from flask_mysqldb import MySQL
from itsdangerous import URLSafeTimedSerializer, SignatureExpired
from werkzeug.utils import secure_filename
import time

#summarization part
import sys
sys.path.insert(0, 'C:/Users/hp/Documents/PROJECT/blog_mining/summarizer')
from summarizer import lsa_summarizer
from lsa_summarizer import LsaSummarizer
import nltk
from nltk.corpus import stopwords

#summarize
def summarize(text):
    q=len(text.split("."))-1
    p=q
    if(q>=8):
        if(q>=70):
            q-=45
        elif(q>=60):
            q-=35
        elif(q>=50):
            q-=30
        elif(q>=40):
            q-=25
        elif(q>=30):
            q-=17
        elif(q>=20):
            q-=12
        elif(q>=12):
            q-=6
        elif(q>=8):
            q-=2
        nltk.download("punkt", quiet=True)
        nltk.download("stopwords", quiet=True)
        summarizer = LsaSummarizer()
        stopWords = stopwords.words('english')
        summarizer.stop_words = stopWords
        summary =summarizer(text,q)
        return " ".join(summary)
```

```
        #return str(p)+" "+str(q)
    else:
        return text

def content_morphe(text):
    txt=""
    p=text.split(".")
    if(len(p)!=1):
        p=p[:-1]
    r=""
    for i in p:
        r=""
        if("'" in i):
            for j in i:
                if(j=="'"):
                    r+=j+"'"
                else:
                    r+=j
        else:
            r+=i
        if(i[0]==' '):
            txt+=r+"."
        else:
            txt+=" "+r+"."
    return txt

LETTERS_SMALLS = 'abcdefghijklmnopqrstuvwxyz'
LETTERS_CAPS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
NUMBERS = '0123456789'

#Encryption
def encrypt(message, key):
    encrypted = ''
    key1=key
    if(key1>10):
        key1=key%10
    if(key>26):
        key=key%26
    for chars in message:
        if chars in LETTERS_SMALLS:
            num = LETTERS_SMALLS.find(chars)
            num =num+key
            if(num>=26):
                num=num%26
            encrypted +=  LETTERS_SMALLS[num]
        elif chars in LETTERS_CAPS:
            num = LETTERS_CAPS.find(chars)
            num =(num+key)
            if(num>=26):
                num=num%26
```

```
            encrypted +=  LETTERS_CAPS[num]
        elif chars in NUMBERS:
            num = NUMBERS.find(chars)
            num =num+key1
            if(num>=10):
                num=num%10
            encrypted +=  NUMBERS[num]
        else:
            encrypted+=chars
    return encrypted

#Decryption
def decrypt(message, key):
    decrypted = ''
    key1=key
    if(key1>10):
        key1=key%10
    if(key>26):
        key=key%26
    for chars in message:
        if chars in LETTERS_SMALLS:
            num = LETTERS_SMALLS.find(chars)
            num =num-key
            decrypted +=  LETTERS_SMALLS[num]
        elif chars in LETTERS_CAPS:
            num = LETTERS_CAPS.find(chars)
            num =num-key
            decrypted +=  LETTERS_CAPS[num]
        elif chars in NUMBERS:
            num = NUMBERS.find(chars)
            num =num-key1
            decrypted +=  NUMBERS[num]
        else:
            decrypted+=chars
    return decrypted

UPLOAD_FOLDER = 'C:/Users/hp/Documents/PROJECT/Blog_mining/static'

s = URLSafeTimedSerializer('Thisisasecret!')

app = Flask(__name__)
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'blog_mining'
mysql = MySQL(app)
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'lhostadlogin127@gmail.com'
app.config['MAIL_PASSWORD'] = ''
```

```
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True

mail= Mail(app)
email="
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'


@app.route("/")
def index():
    try:
        cur=mysql.connection.cursor()
        cur.execute("SELECT * FROM post ORDER BY NUM_VIEWS DESC LIMIT 4")
        records=cur.fetchall()
        featured=[]
        i=0
        even=[]
        odd=[]
        for row in records:
            content=row[7][:115]+str("...")
            if(i%2==0):

even.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row
[4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
            else:

odd.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row[
4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
            i+=1
        featured.extend(even)
        featured.extend(odd)
        cur.execute("SELECT * FROM post ORDER BY ID DESC LIMIT 6")
        records=cur.fetchall()
        i=0
        all_stories=[]
        first=[]
        second=[]
        third=[]
        for row in records:
            content=row[7][:150]+str("...")
            if(i==0):

first.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row[
4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
                i+=1
            elif(i==1):

second.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':ro
w[4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
                i+=1
```

```
            elif(i==2):

third.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row
[4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
                i=0
        all_stories.extend(first)
        all_stories.extend(second)
        all_stories.extend(third)
        cur.execute("SELECT CATEGORY FROM post WHERE ID="+str(1))
        record=cur.fetchone()
        return
render_template("index.html",featured=featured,all_stories=all_stories,token=record[0])
    except:
        return redirect(url_for('handle_exception'))


@app.route("/search",methods=['GET','POST'])
def search():
    if request.method=='POST':
        search_content=request.form['search_content']
        search_content=search_content.lower()
        if(search_content==""):
            cur=mysql.connection.cursor()
            cur.execute("SELECT CATEGORY FROM post WHERE ID="+str(1))
            record=cur.fetchone()
            return
render_template("search.html",search_content=0,token=record[0],search_contents=search_co
ntent)
        tic = time.perf_counter()
        lst=search_content.split(" ")
        cur=mysql.connection.cursor()
        cur.execute("SELECT ID,CATEGORY,TITLE,CONTENT FROM post ORDER BY ID
DESC")
        records=cur.fetchall()
        d={}
        for row in records:
            st=""
            p=0
            st+=row[1].lower()+" "+row[2].lower()+" "+row[3].lower()
            for i in lst:
                p+=st.count(i)
            d[row[0]]=p
        sorted_values=sorted(d.values())
        sorted_values=sorted_values[::-1]
        sorted_dict={}
        for i in sorted_values:
            for k in d.keys():
                if(i==0):
                    break
                if d[k]==i:
```

```
                sorted_dict[k]=d[k]
                d[k]=-1
                break
        j=0
        search_list=[]
        even=[]
        odd=[]
        for i in sorted_dict:
            cur.execute("SELECT * FROM post WHERE ID="+str(i))
            row=cur.fetchone()
            content=row[7][:115]+str("...")
            if(j%2==0):

even.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row
[4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
            else:

odd.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row[
4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
            j+=1
        search_list.extend(even)
        search_list.extend(odd)
        cur.execute("SELECT CATEGORY FROM post WHERE ID="+str(1))
        record=cur.fetchone()
        toc = time.perf_counter()
        print(toc-tic)
        if(search_list==[]):
            return
render_template("search.html",search_content=0,token=record[0],search_contents=search_co
ntent)
        else:
            return
render_template("search.html",search_list=search_list,token=record[0],search_contents=searc
h_content)


@app.route("/view_all")
def view_all():
    try:
        cur=mysql.connection.cursor()
        cur.execute("SELECT * FROM post ORDER BY ID DESC")
        records=cur.fetchall()
        all_stories=[]
        i=0
        even=[]
        odd=[]
        for row in records:
            content=row[7][:115]+str("...")
            if(i%2==0):
```

```
even.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row
[4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
            else:

odd.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row[
4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
            i+=1
        all_stories.extend(even)
        all_stories.extend(odd)
        cur.execute("SELECT CATEGORY FROM post WHERE ID="+str(1))
        record=cur.fetchone()
        return render_template("view_all.html",all_stories=all_stories,token=record[0])
    except:
        return redirect(url_for('handle_exception'))


@app.route("/post/<token>")
def post(token):
    try:
        cur=mysql.connection.cursor()
        cur.execute("SELECT * FROM post WHERE ID="+str(token))
        record=cur.fetchone()
        post={}
        category=record[1]
        title=record[2]
        updated_by=record[3]
        updated_on=record[4]
        num_min=record[5]
        num_views=record[6]
        content=record[7]
        photo=record[8]
        num_views=str((int(num_views)+1))
        cur.execute("UPDATE post SET NUM_VIEWS=%s WHERE ID=%s" %( num_views,
str(token) ) )
        mysql.connection.commit()
        cur.execute("SELECT * FROM categories LIMIT 4")
        records=cur.fetchall()
        categories=[]
        for row in records:
            categories.append(row[0])
        cur.execute("SELECT * FROM post ORDER BY NUM_VIEWS DESC LIMIT 3")
        records=cur.fetchall()
        lst=[]
        for row in records:
            content_vary=row[7]+str("...")

lst.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row[4
],'num_min':row[5],'num_views':row[6],'content':content_vary,'photo':row[8]})
        return
render_template("post.html",category=category,title=title,updated_by=updated_by,updated_o
```

```
n=updated_on,num_min=num_min,num_views=num_views,content=content,photo=photo,to
ken=record[0],categories=categories,suggestive_posts=lst)
   except:
      return redirect(url_for('handle_exception'))


@app.route("/categories/<token>")
def categories(token):
   try:
      cur=mysql.connection.cursor()
      cur.execute("SELECT * FROM categories")
      records=cur.fetchall()
      categories=[]
      for row in records:
         if(row[0]!=token):
            categories.append(row[0])
      cur.execute("SELECT * FROM post WHERE CATEGORY='%s'" %(str(token),))
      records=cur.fetchall()
      category=[]
      i=0
      even=[]
      odd=[]
      for row in records:
         content=row[7][:115]+str("...")
         if(i%2==0):

even.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row
[4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
         else:

odd.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row[
4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
         i+=1
      category.extend(even)
      category.extend(odd)
      cur.execute("SELECT CATEGORY FROM post WHERE ID="+str(1))
      record=cur.fetchone()
      if(category==[]):
         return
render_template("categories.html",search_content=0,token=token,categories=categories,categ
ory=category,record=record[0])
      else:
         return
render_template("categories.html",token=token,categories=categories,category=category,reco
rd=record[0])
   except:
      return redirect(url_for('handle_exception'))


@app.route("/search_category/<token>",methods=['GET','POST'])
```

```
def search_category(token):
    try:
        if request.method=='POST':
            cur=mysql.connection.cursor()
            cur.execute("SELECT ID,TITLE,CONTENT FROM post WHERE CATEGORY='%s'
ORDER BY ID DESC" %(str(token),))
            records=cur.fetchall()
            search_content=request.form['search_category']
            if(search_content==""):
                cur.execute("SELECT * FROM categories")
                records=cur.fetchall()
                categories=[]
                for row in records:
                    if(row[0]!=token):
                        categories.append(row[0])
                cur.execute("SELECT CATEGORY FROM post WHERE ID="+str(1))
                record=cur.fetchone()
                return
render_template("search_category.html",search_content=0,token=token,categories=categories
,record=record)
            search_content=search_content.lower()
            lst=search_content.split(" ")
            d={}
            for row in records:
                st=""
                p=0
                st+=row[1].lower()+" "+row[2].lower()
                for i in lst:
                    p+=st.count(i)
                d[row[0]]=p
            sorted_values=sorted(d.values())
            sorted_values=sorted_values[::-1]
            sorted_dict={}
            for i in sorted_values:
                for k in d.keys():
                    if(i==0):
                        break
                    if d[k]==i:
                        sorted_dict[k]=d[k]
                        d[k]=-1
                        break
            j=0
            search_list=[]
            even=[]
            odd=[]
            for i in sorted_dict:
                cur.execute("SELECT * FROM post WHERE ID="+str(i))
                row=cur.fetchone()
                content=row[7][:115]+str("...")
                if(j%2==0):
```

```
even.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row
[4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
            else:

odd.append({'id':row[0],'category':row[1],'title':row[2],'updated_by':row[3],'updated_on':row[
4],'num_min':row[5],'num_views':row[6],'content':content,'photo':row[8]})
            j+=1
        search_list.extend(even)
        search_list.extend(odd)
        categories=[]
        cur=mysql.connection.cursor()
        cur.execute("SELECT * FROM categories")
        records=cur.fetchall()
        categories=[]
        for row in records:
            if(row[0]!=token):
                categories.append(row[0])
        cur.execute("SELECT CATEGORY FROM post WHERE ID="+str(1))
        record=cur.fetchone()
        if(search_list==[]):
            return
render_template("search_category.html",search_content=0,search_list=search_list,token=toke
n,categories=categories,record=record)
        else:
            return
render_template("search_category.html",search_list=search_list,token=token,categories=cate
gories,record=record)
    except:
        return redirect(url_for('handle_exception'))


@app.route("/author",methods=['GET', 'POST'])
def author():
    if(request.method=='GET'):
        cur=mysql.connection.cursor()
        cur.execute("SELECT CATEGORY FROM post WHERE ID="+str(1))
        record=cur.fetchone()
        return render_template("author.html",token=record)
    cur=mysql.connection.cursor()
    name=request.form['name']
    mesg=request.form['message']
    sub=request.form['subject']
    email=request.form['email']
    msg=Message("Contact
us",sender='lhostadlogin127@gmail.com',recipients=['lhostadlogin127@gmail.com'])
    msg.body="Name: "+str(name)+"\nGenerated from: "+email+"\nSubject:
"+sub+"\nMessage: "+str(mesg)
    mail.send(msg)
    msg=Message(sub,sender='lhostadlogin127@gmail.com',recipients=[str(email)])
```

```
    msg.body="Hi "+str(name)+"\nSubject: "+sub+"\nMessage: "+mesg+"\nWe would be in
contact with you soonerr!!"
    mail.send(msg)
    cur.execute("SELECT CATEGORY FROM post WHERE ID="+str(1))
    record=cur.fetchone()
    return render_template("author.html",token=record[0])


@app.route("/login",methods=['GET','POST'])
def login():
    try:
        if request.method=='GET':
            return render_template("LOGIN.html")
        uname=request.form['username']
        pwd=request.form['password']
        cur=mysql.connection.cursor()
        cur.execute("SELECT USERNAME,PASSWORD,EID FROM login")
        records=cur.fetchall()
        flag=0
        for row in records:
            if(row[0]==uname):
                flag=1
                p=decrypt(row[1],row[2])
                if(p==pwd):
                    flag=2
                    break
        msg=""
        if(flag==2):
            return redirect(url_for("blog_info",token=uname))
        elif(flag==1):
            msg="INCORRECT  PASSWORD"
        else:
            msg="INCORRECT USERNAME/PASSWORD"
        flash(msg)
        return render_template("login.html")
    except:
        return redirect(url_for('handle_exception'))



@app.route("/update_content/<token>",methods=['GET','POST'])
def update_content(token):
    if request.method=='GET':
        cur=mysql.connection.cursor()
        cur.execute("SELECT USERNAME,EMAIL FROM login WHERE USERNAME='%s'"
%(str(token),))
        records=cur.fetchone()
        return render_template("update_content.html",username=records[0],email=records[1])
    cur=mysql.connection.cursor()
    cur.execute("SELECT ID FROM post ORDER BY ID DESC LIMIT 1")
```

```
    records=cur.fetchone()
    if(records is None):
        id_post=1
    else:
        id_post=str(int(records[0])+1)
    category=request.form['category']
    title=request.form['title']
    content=content_morphe(request.form['content'])
    summarized_content=summarize(content)
    num_lines=str(len(content.split("."))-1)
    smry_num_lines=str(len(summarized_content.split("."))-1)
    num_words=str(len(content.split(" "))-1)
    smry_num_words=str(len(summarized_content.split(" "))-1)
    updated_by=str(token)
    today = date.today()
    updated_on=str(today.strftime("%y-%m-%d"))
    p=(len(summarized_content.split('.'))-1)
    l=len(summarized_content.split(" "))
    q=l%60
    if(q>0):
        num_min=str((l//60)+1)
    else:
        num_min=str(l//60)
    p=(len(content.split('.'))-1)
    l=len(content.split(" "))
    q=l%60
    if(q>0):
        actual_num_min=str((l//60)+1)
    else:
        actual_num_min=str(l//60)
    num_views=str(0)
    file = request.files['file']
    if file:
        filename = secure_filename(file.filename)
        p=file.filename
        lst=list(p.split("."))
        file_name=str(id_post)+"."+str(lst[-1])
        file.save(os.path.join(UPLOAD_FOLDER,file_name))
#app.config['UPLOAD_FOLDER']
    photo=str(file_name)
    cur.execute("SELECT * FROM categories")
    records=cur.fetchall()
    flag=0
    for row in records:
        if(row[0]==category):
            flag=1
            break
    if(flag==0):
        cur.execute("INSERT INTO categories VALUES('%s')" %(category,))
        mysql.connection.commit()
```

```
    cur.execute("INSERT INTO post VALUES('%s','%s','%s','%s','%s','%s','%s','%s','%s')"
%(id_post,category,title,updated_by,updated_on,num_min,num_views,summarized_content,p
hoto))
    mysql.connection.commit()
    #cur.execute("INSERT INTO analyze_details
VALUES('%s','%s',%s','%s','%s','%s','%s','%s','%s')"
%(id_post,title,content,num_lines,smry_num_lines,num_words,smry_num_words,actual_nu
m_min,num_min))
    cur.execute("INSERT INTO analyze_details
VALUES('%s','%s','%s','%s','%s','%s','%s','%s','%s')"
%(id_post,title,content,num_lines,smry_num_lines,num_words,smry_num_words,actual_nu
m_min,num_min))
    mysql.connection.commit()
    flash("Sucessfully updated!!")
    cur.execute("SELECT EMAIL FROM login WHERE USERNAME='%s'" %(str(token),))
    records=cur.fetchone()
    return render_template("update_content.html",username=token,email=records[0])
    #return str(id_post)+" "+title+" "+content+" "+num_lines+" "+smry_num_lines+"
"+num_words+" "+smry_num_words+" "+actual_num_min+" "+num_min


@app.route("/reset",methods=['GET','POST'])
def reset():
    try:
        if request.method=='GET':
            return render_template("reset.html")
        email=request.form['email']
        cur=mysql.connection.cursor()
        cur.execute("SELECT EMAIL FROM login")
        records=cur.fetchall()
        flag=0
        for row in records:
            if(row[0]==email):
                flag=1
                break
        cur.close()
        if(flag==1):
            token = s.dumps(email, salt='email-confirm')
            msg=Message('Password Confirmation
mail',sender='lhostadlogin127@gmail.com',recipients=[email])
            msg.body = "Please complete reset process within an hour, rather token expires.\nYour
link: "+"http://127.0.0.1:5000/reset_email/"+token
            mail.send(msg)
            flash("Please check your mail!!!!")
            return redirect(url_for('reset'))
        else:
            flash("Email-Id isnot Registered!!")
            return redirect(url_for('reset'))
    except:
        return redirect(url_for('handle_exception'))
```

```
@app.route('/reset_email/<token>',methods=['GET', 'POST'])
def reset_email(token):
    try:
        email = s.loads(token, salt='email-confirm', max_age=3600)
        if request.method=='GET':
            return render_template("reset_email.html",token=token)
        cur=mysql.connection.cursor()
        cur.execute("SELECT EID FROM login WHERE EMAIL=%s",(email,))
        record=cur.fetchone()
        new_pwd=request.form['pwd']
        enc=encrypt(new_pwd,record[0])
        val=""
        cur=mysql.connection.cursor()
        cur.execute("UPDATE login SET PASSWORD=%s WHERE EMAIL=%s",(enc,email))
        mysql.connection.commit()
        cur.close()
        flash("Password successfully updated!!!!!")
        return redirect(url_for('login'))
    except SignatureExpired:
        return '<h1>The token is expired!</h1>'


@app.route("/profile_info/<token>",methods=['GET','POST'])
def profile_info(token):
    if request.method=='GET':
        cur=mysql.connection.cursor()
        cur.execute("SELECT USERNAME,EMAIL,CONTACT_NUMBER,CITY,STATE
FROM login WHERE USERNAME='%s'" %(str(token),))
        record=cur.fetchone()
        return
render_template("profile_info.html",username=record[0],email=record[1],contact_number=re
cord[2],city=record[3],state=record[4],token=token)
    cur=mysql.connection.cursor()
    contact_number=request.form['contact_number']
    city=request.form['city']
    state=request.form['state']
    cur.execute("UPDATE login SET CONTACT_NUMBER='%s',CITY='%s',STATE='%s'
WHERE USERNAME='%s'" %(str(contact_number),city,state,token))
    mysql.connection.commit()
    cur.close()
    flash("Personal info successfully updated!!")
    return redirect(url_for('profile_info',token=token))


@app.route("/blog_info/<token>",methods=['GET','POST'])
def blog_info(token):
    if request.method=='POST':
        title=request.form['title']
```

```
        cur=mysql.connection.cursor()
        cur.execute("DELETE FROM post WHERE TITLE='%s'" %(title,))
        mysql.connection.commit()
        cur.execute("DELETE FROM analyze_details WHERE TITLE='%s'" %(title,))
        mysql.connection.commit()
        cur.execute("SELECT TITLE FROM post WHERE UPDATED_BY='%s'" %(token,))
        records=cur.fetchall()
        lst=[]
        c=0
        for row in records:
            lst.append(row[0])
            c+=1
        cur=mysql.connection.cursor()
        cur.execute("SELECT EMAIL FROM login WHERE USERNAME='%s'" %(str(token),))
        records=cur.fetchone()
        return
render_template("blog_info.html",lst=lst,num_posts=c,token=token,email=records[0])
    cur=mysql.connection.cursor()
    cur.execute("SELECT TITLE FROM post WHERE UPDATED_BY='%s'" %(token,))
    records=cur.fetchall()
    lst=[]
    c=0
    for row in records:
        lst.append(row[0])
        c+=1
    cur=mysql.connection.cursor()
    cur.execute("SELECT EMAIL FROM login WHERE USERNAME='%s'" %(str(token),))
    records=cur.fetchone()
    return
render_template("blog_info.html",lst=lst,num_posts=c,token=token,email=records[0])


@app.route("/blog_info_edit/<token>",methods=['GET','POST'])
def blog_info_edit(token):
    if request.method=='GET':
        st=token.split('_')
        cur=mysql.connection.cursor()
        cur.execute("SELECT
CATEGORY,TITLE,CONTENT,NUM_VIEWS,PHOTO,UPDATED_ON FROM post
WHERE TITLE='%s'" %(st[1],))
        records=cur.fetchone()
        return
render_template("blog_info_edit.html",token=st[0],category=records[0],title=records[1],conte
nt=records[2],num_views=records[3],photo=records[4],updated_on=records[5],token1=st[0]+
'_'+records[1])
    st=token.split('_')
    cur=mysql.connection.cursor()
    cur.execute("SELECT ID,photo FROM post WHERE TITLE='%s'" %(st[1]))
    records=cur.fetchone()
    id_post=records[0]
```

```
    category=request.form['category']
    title=request.form['title']
    content=content_morphe(request.form['content'])
    summarized_content=summarize(content)
    num_lines=str(len(content.split("."))-1)
    smry_num_lines=str(len(summarized_content.split("."))-1)
    num_words=str(len(content.split(" "))-1)
    smry_num_words=str(len(summarized_content.split(" "))-1)
    today = date.today()
    updated_on=str(today.strftime("%y-%m-%d"))
    p=(len(summarized_content.split('.'))-1)
    l=len(summarized_content.split(" "))
    q=l%60
    if(q>0):
        num_min=str((l//60)+1)
    else:
        num_min=str(l//60)
    p=(len(content.split('.'))-1)
    l=len(content.split(" "))
    q=l%60
    if(q>0):
        actual_num_min=str((l//60)+1)
    else:
        actual_num_min=str(l//60)
    num_views=str(0)
    file = request.files['file']
    file_name=""
    if file:
        filename = secure_filename(file.filename)
        p=file.filename
        lst=list(p.split("."))
        file_name=str(id_post)+"."+str(lst[-1])
        file.save(os.path.join(UPLOAD_FOLDER,file_name))
#app.config['UPLOAD_FOLDER']
        photo=str(file_name)
    else:
        photo=str(records[1])
    cur.execute("SELECT * FROM categories")
    records=cur.fetchall()
    flag=0
    for row in records:
        if(row[0]==category):
            flag=1
            break
    if(flag==0):
        cur.execute("INSERT INTO categories VALUES('%s')" %(category,))
        mysql.connection.commit()
    cur.execute("UPDATE post SET CATEGORY='%s', TITLE='%s', UPDATED_ON='%s',
NUM_MIN='%s', NUM_VIEWS='%s', CONTENT='%s',PHOTO='%s' WHERE ID='%s' "
%(str(category),str(title),str(updated_on),str(num_min),str(num_views),str(summarized_conte
```

```
nt),str(photo),str(id_post)))
    mysql.connection.commit()
    cur.execute("UPDATE analyze_details SET TITLE='%s',ORIGINAL_CONTENT='%s',
NUM_LINES_BEFORE='%s',
NUM_LINES_AFTER='%s',NUM_WORDS_BEFORE='%s',NUM_WORDS_AFTER='%s',
MIN_READ_BEFORE='%s',MIN_READ_AFTER='%s' WHERE ID='%s' "
%(title,content,num_lines,smry_num_lines,num_words,smry_num_words,actual_num_min,n
um_min,id_post))
    mysql.connection.commit()
    flash("Sucessfully updated!!")
    return redirect(url_for('blog_info_edit',token=st[0]+"_"+title))


@app.route("/documentation")
def documentation():
    return render_template("documentation.html")

if __name__ == '__main__':
   app.run(debug = True)
```

lsa_summarizer.py:

```
from __future__ import absolute_import
from __future__ import division, print_function, unicode_literals
import math
import numpy
import nltk

import sys
sys.path.insert(0, 'C:/Users/hp/Documents/PROJECT/blog_mining/summarizer')
from summarizer import base_summarizer

from warnings import warn
from nltk.tokenize import sent_tokenize, word_tokenize
from numpy.linalg import svd as singular_value_decomposition
from base_summarizer import BaseSummarizer
from nltk.corpus import stopwords

class LsaSummarizer(BaseSummarizer):
    MIN_DIMENSIONS = 3
    REDUCTION_RATIO = 1/1

    _stop_words = list(stopwords.words('english'))

    @property
    def stop_words(self):
        return self._stop_words

    @stop_words.setter
    def stop_words(self, words):
```

```
        self._stop_words = words

    def __call__(self, document, sentences_count):

        dictionary = self._create_dictionary(document)

        if not dictionary:
            return ()

        sentences = sent_tokenize(document)

        matrix = self._create_matrix(document, dictionary)
        matrix = self._compute_term_frequency(matrix)
        u, sigma, v = singular_value_decomposition(matrix, full_matrices=False)

        ranks = iter(self._compute_ranks(sigma, v))
        return self._get_best_sentences(sentences, sentences_count,
            lambda s: next(ranks))

    def _create_dictionary(self, document):
        """Creates mapping key = word, value = row index"""

        words = word_tokenize(document)
        words = tuple(words)

        words = map(self.normalize_word, words)

        unique_words = frozenset(w for w in words if w not in self._stop_words)

        return dict((w, i) for i, w in enumerate(unique_words))

    def _create_matrix(self, document, dictionary):
        """
        Creates matrix of shape where cells
        contains number of occurences of words (rows) in senteces (cols).
        """
        sentences = sent_tokenize(document)
        words_count = len(dictionary)
        sentences_count = len(sentences)
        if words_count < sentences_count:
            message = (
                "Number of words (%d) is lower than number of sentences (%d). "
                "LSA algorithm may not work properly."
            )
            warn(message % (words_count, sentences_count))

        matrix = numpy.zeros((words_count, sentences_count))
        for col, sentence in enumerate(sentences):
            words = word_tokenize(sentence)
            for word in words:
```

```
            # only valid words is counted (not stop-words, ...)
            if word in dictionary:
                row = dictionary[word]
                matrix[row, col] += 1

        return matrix

    def _compute_term_frequency(self, matrix, smooth=0.4):
        """
        Computes TF metrics for each sentence (column) in the given matrix and  normalize
        the tf weights of all terms occurring in a document by the maximum tf in that document
        according to ntf_{t,d} = a + (1-a)\frac{tf_{t,d}}{tf_{max}(d)^{'}}.

        The smoothing term $a$ damps the contribution of the second term - which may be viewed
        as a scaling down of tf by the largest tf value in $d$
        """
        assert 0.0 <= smooth < 1.0

        max_word_frequencies = numpy.max(matrix, axis=0)
        rows, cols = matrix.shape
        for row in range(rows):
            for col in range(cols):
                max_word_frequency = max_word_frequencies[col]
                if max_word_frequency != 0:
                    frequency = matrix[row, col]/max_word_frequency
                    matrix[row, col] = smooth + (1.0 - smooth)*frequency

        return matrix

    def _compute_ranks(self, sigma, v_matrix):
        assert len(sigma) == v_matrix.shape[0]

        dimensions = max(LsaSummarizer.MIN_DIMENSIONS,
            int(len(sigma)*LsaSummarizer.REDUCTION_RATIO))
        powered_sigma = tuple(s**2 if i < dimensions else 0.0
            for i, s in enumerate(sigma))

        ranks = []

        for column_vector in v_matrix.T:
            rank = sum(s*v**2 for s, v in zip(powered_sigma, column_vector))
            ranks.append(math.sqrt(rank))

        return ranks
```

base_summarizer.py:

```
from operator import attrgetter
from collections import namedtuple
import sys
```

```
sys.path.insert(0, 'C:/Users/hp/Documents/PROJECT/blog_mining/summarizer')
from summarizer import utils
from utils import ItemsCount

SentenceInfo = namedtuple("SentenceInfo", ("sentence", "order", "rating",))

class BaseSummarizer(object):

    def __call__(self, document, sentences_count):
        raise NotImplementedError("This method should be overriden in subclass")

    @staticmethod
    def normalize_word(word):
        return word.lower()

    @staticmethod
    def _get_best_sentences(sentences, count, rating, *args, **kwargs):
        rate = rating
        if isinstance(rating, dict):
            assert not args and not kwargs
            rate = lambda s: rating[s]

        infos = (SentenceInfo(s, o, rate(s, *args, **kwargs))
            for o, s in enumerate(sentences))

        # sort sentences by rating in descending order
        infos = sorted(infos, key=attrgetter("rating"), reverse=True)
        # get `count` first best rated sentences
        if not isinstance(count, ItemsCount):
            count = ItemsCount(count)
        infos = count(infos)
        # sort sentences by their order in document
        infos = sorted(infos, key=attrgetter("order"))

        return tuple(i.sentence for i in infos)
```

utils.py:

```
class ItemsCount(object):
    def __init__(self, value):
        self._value = value

    def __call__(self, sequence):
        if isinstance(self._value, (bytes, str,)):
            if self._value.endswith("%"):
                total_count = len(sequence)
                percentage = int(self._value[:-1])
                # at least one sentence should be chosen
                count = max(1, total_count*percentage // 100)
                return sequence[:count]
```

```
        else:
            return sequence[:int(self._value)]
    elif isinstance(self._value, (int, float)):
        return sequence[:int(self._value)]
    else:
        ValueError("Unsuported value of items count '%s'." % self._value)

def __repr__(self):

    return to_string("<ItemsCount: %r>" % self._value)
```

7.2 BIBLIOGRAPHY

[1]. H.P. Edmunson. 1969. New methods in automatic extracting. Journal of the ACM, 16(2):264–285.

[2]. G. Erkan and D. Radev. 2004. Lexpagerank: Prestige in multi-document text Summarization. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), Barcelona, Spain, July.

[3]. W. Li, W. Li, B. Li, Q. Chen, and M. Wu. 2005. The Hong Kong Polytechnic University at DUC 2005. In Proceedings of the Document Understanding Conference (DUC 2005), Vancouver, Canada.

[4]. F. Wolf and E. Gibson. 2004. Paragraph-, word-, and coherence-based approaches to sentence ranking: A comparison of algorithm and human performance. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2004), Barcelona, Spain, July

[5]. S. Wan and K. McKeown. 2004. Generating overview summaries of ongoing email thread discussions. In Proceedings of the 20th International Conference on Computational Linguistics, Geneva, Switzerland.

[6]. L. Zhou and E. Hovy. 2003. A Web-trained extraction summarization system. In Proceedings of Human Language Technology Conference (HLT-NAACL 2003), Edmonton, Canada, May.

[7]. L. Zhou and E. Hovy. 2005. Digesting virtual ”geek” culture: The summarization of technical internet relay chats. In Proceedings of Association for Computational Linguistics (ACL 2005), Ann Arbor.

[8]. L. Zhuang, F. Jing, and X.Y. Zhu. 2006. Movie review mining and summarization. In Proceedings of the ACM international conference on Information and knowledge management (CIKM 2006), Arlington, Virginia.

[9]. Mehwish Aziz, Muhammad Rafi, Sentence-Based Semantic Similarity Measure for Blog Posts, 6th International Conference on Digital Content, Multimedia Technology and its Applications, IDC 2010, Aug 2010, IEEE Seoul.

[10]. Mehwish Aziz, Muhammad Rafi, Identifying Influential Bloggers using Blogs Semantics, Frontier of Information Technology, Dec 2010, ACM Islamabad.

[11]. Gilbert, E.; Bergstrom, T.; Karahalios, K.;, "Blogs are Echo Chambers: Blogs are Echo Chambers," System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on, vol., no., pp.1-10, 5-8 Jan. 2009 doi: 10.1109/HICSS.2009.91

[12]. Kushal Dave, Steve Lawrence, David M. Pennock. 2003. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In Proceedings of 12th international conference on World Wide Web (WWW '03). ACM, New York, NY, USA, 519-528


CERTIFICATE
OF PUBLICATION
IJIRCCE
International Journal of Innovative Research in Computer and Communication Engineering
Website: www.ijircce.com Email: ijircce@gmail.com
This is hereby Awarding this Certificate to
M. SRUTHI
B.Tech Student, Department of IT, Vasireddy Venkatadri Institute of Technology, Guntur, India
Published a paper entitled
Blog Mining and Emotion Argumentation
in IJIRCCE, Volume 10, Issue 5, May 2022
Impact Factor 8.165
e-ISSN: 2320-9801
p-ISSN: 2320-9798
ISSN
INNO SPACE
SJIF Scientific Journal Impact Factor
Editor-in-Chief


CERTIFICATE
OF PUBLICATION
IJIRCCE
International Journal of Innovative Research in Computer and Communication Engineering
Website: www.ijircce.com Email: ijircce@gmail.com
This is hereby Awarding this Certificate to
K. VEENA MADHURI
B.Tech Student, Department of IT, Vasireddy Venkatadri Institute of Technology, Guntur, India
Published a paper entitled
Blog Mining and Emotion Argumentation
in IJIRCCE, Volume 10, Issue 5, May 2022
Impact Factor 8.165
e-ISSN: 2320-9801
p-ISSN: 2320-9798
ISSN
INNO SPACE
SJIF Scientific Journal Impact Factor
Editor-in-Chief


CERTIFICATE
OF PUBLICATION
IJIRCCE
International Journal of Innovative Research in Computer and Communication Engineering
Website: www.ijircce.com Email: ijircce@gmail.com
This is hereby Awarding this Certificate to
M. SRAVYA
B.Tech Student, Department of IT, Vasireddy Venkatadri Institute of Technology, Guntur, India
Published a paper entitled
Blog Mining and Emotion Argumentation
in IJIRCCE, Volume 10, Issue 5, May 2022
Impact Factor 8.165
e-ISSN: 2320-9801
p-ISSN: 2320-9798
ISSN
INTERNATIONAL STANDARD SERIAL NUMBER
INNO SPACE
SJIF Scientific Journal Impact Factor
Editor-in-Chief


CERTIFICATE
OF PUBLICATION
IJIRCCE
International Journal of Innovative Research in Computer and Communication Engineering
Website: www.ijircce.com Email: ijircce@gmail.com
This is hereby Awarding this Certificate to
K. SARASWATHI
B.Tech Student, Department of IT, Vasireddy Venkatadri Institute of Technology, Guntur, India
Published a paper entitled
Blog Mining and Emotion Argumentation
in IJIRCCE, Volume 10, Issue 5, May 2022
Impact Factor 8.165
e-ISSN: 2320-9801
p-ISSN: 2320-9798
ISSN
INTERNATIONAL STANDARD SERIAL NUMBER
INNO SPACE
SJIF Scientific Journal Impact Factor
Editor-in-Chief

BLOG MINING AND EMOTION ARGUMENTATION

Ms. K. Vineela[1], M. Sruthi[2], M. Sravya[3], K. Saraswathi[4], K. Veena Madhuri[5]

Assistant Professor, B.Tech Students

DEPARTMENT OF INFORMATION TECHNOLOGY

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Abstract –

Blogs are perhaps widely used by the internet users due to their ability to disseminate information and present their ideas on various topics. These blogs have increasingly become an important information source for the users' ideas. So, we are providing a platform to Blogger's and Reader's, where the blogger will continually update new content or information in the Blog and readers will check them frequently and gain knowledge or information from the blogs. These subjective information in blogs helps in understanding a blogger's views and observations about various topics. So, we created a website providing access for users and bloggers, where bloggers have a separate account in the website, they can login and update/write the content and readers can go to the home page directly to view the blogs of bloggers. Here we are creating a Techie Blog website.

Blog Mining and Emotion Argumentation provides a capability of processing large amount of text data effectively, it can be a valuable method for gaining insights into a given topic. Due to its capability of processing large amount of text data effectively, the overloaded content is summarized and the user can easily deluge huge amount of information in a shorter span and also allowing quicker search results.

The steps involved here are:

1. Pre-processing of data
2. Thus, pre-processed data is updated by blogger
3. The info updated is summarized (involves text summarization)
4. Readers browse the content and access necessary blog where deluge content is summarized and user gain insights about a blog in shorter span and also having fast search results.

1. Introduction –

A blog (a shortened version of "weblog") is an online journal or informational website displaying information in reverse chronological order, with the latest posts appearing first, at the top. It is an informational website, often informal diary-style text entries, and acting as a platform where a writer or a group of writers share their views on an individual subject. It is similar to an online journal where an individual, group, or corporation presents a record of activities, thoughts, or beliefs. In general terminology blog is explained as a regularly updated website or web page, typically run by an individual or small group, that is written in an informal or conversational style, consisting of series of posts where posts are archived, and are usually sorted into categories. It is similar to a newspaper in that it publishes new items on a regular basis and keeps the older ones up to date. Bloggers identify the sentiments, both positive and negative opinions about the topic to understand and present public views in detail. Readers can browse these categories through the blog to read older entries. It does typically involve searching and analysing blogs in order to generate additional insights and acts as an information source for the user's ideas. Blog Mining and Emotion Argumentation provides a capability of processing large amount of text data effectively. This study of blog mining is used to analyse and search the online blog posts relevant contents in a quite simpler fashion.

2. Aim and Scope, Goals and Objectives –

Aim and Scope –

Blog mining and Emotion Argumentation allows to extract the huge amount of corpus and allows to view it in as condensed state. An individual user diverges respective contents of blog, the encapsulated content allows to digest the huge info in a compact way, consuming minimal time impact. The processing of contents via browsing reduces the actual time consumption to lower. The series of blogs when updated via from the end of the user, which there by provides a summarized view updation to end of database. The main constraints involved are blog corpus source, depictive info of a blog and representative contents, Machine Learning view impact to overcome the complication. Argumentation is the process by which arguments are constructed and handled. Argumentation constitutes a major component of human intelligence. Argumentation is a collection of propositions, all of which are premises except, at most one, which is a conclusion.

Goals and Objectives –

The typical goals of the Blog mining and Emotion Argumentation are to collect the blog corpus and design a system for topic identification and other text processing tasks such as text summarization unit. As the amount of text keeps growing, it becomes increasing difficult for humans to process the deluge of information in the time available. It does result in consumption of huge amount of duration. As a part, the subject propagated within the blogs may not be reachable to end users. Blog Mining overcomes by performing text routing techniques like text summarization. In inclusion of Emotion Argumentation is used for developing, analyzing and categorization the arguments. The overloaded content which is extracted from huge number of posts result huge time to browse as a process of application of summarization provides a faster pace of search.

3. Existing System and Disadvantages –

Existing System –
Blogs are often updated webpages that are being important information sites about a specific topic. Food, fashion, and marketing are just a few of the topics covered by various sorts of blogs. Here the source that we have taken is techie blog. Generally, all posts are archived in blog, and are usually sorted into categories. Readers can browse these categories of a blog to read older entries. It does typically involve searching and analysing blogs in order to generate additional insights and acts as a information source for the user's ideas.

Disadvantages –
1. The overloaded content which is extracted from huge number of posts provides out low search results consuming huge time.
2. As the amount of text keeps growing, it becomes increasing difficult for humans to process the deluge of information in the time available. It does result in consumption of huge amount of duration. As a part of low efficiency, the subject propagated within the blogs may not be reachable to end users.

4. Problem and Proposed Solution –

Problem –
Bloggers are facing their toughest challenge yet and it's called content saturation. There's just too much information to process these days. And when people feel overwhelmed, they react in ways that aren't good for a respective blog. Information overload occurs when a person is exposed to more information than the brain can process at one time. As the amount of on-line text keeps growing, it becomes increasing difficult for humans to process the deluge of information in the time available. The rapid growth of blog documents in web and categorizing search applications based on topics motivates to develop a system that provides identification of the blog documents.

Proposed Solution –
Text summarization is the process of creating a short, accurate, and fluent summary of a longer text document. The main purpose of text summarization is to get the most precise and useful information from a large document and eliminate the irrelevant or less important ones. The process brings out information that is crucial, and also ensures that the meaning of the paragraph stays the same. This helps reduce the time to understand large papers like research articles, without skipping any vital information. The main benefits are makes reading easier, saves time, helps to memorize the information easily, boosts the work rate efficiency. This process allows consumption time to be shorter having quicker search results and emphasizing with Machine learning based text processing tasks such as text summarization unit, and allows to Emotional Argumentation via to find the effect of emotions in the generation of conclusions and evaluate the consistency of emotions from a set of premises to its corresponding conclusion.

5. Methodology

Latent Sematic Analysis (Algo)

The main aim of latent semantic analysis is to create representations of text data in terms of the features and latent features. Latent semantic analysis (LSA) is a mathematical method for computer modelling and simulation of the meaning of words and passages by analysis of representative corpora of natural text. LSA closely approximates many aspects of human language learning and understanding. It supports a variety of applications in information retrieval. The latent sematic analysis consists of two steps:

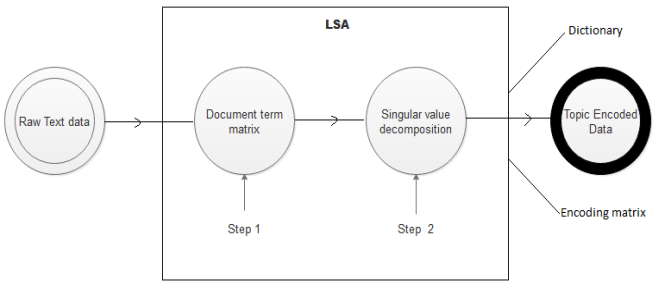


Fig. LSA processing

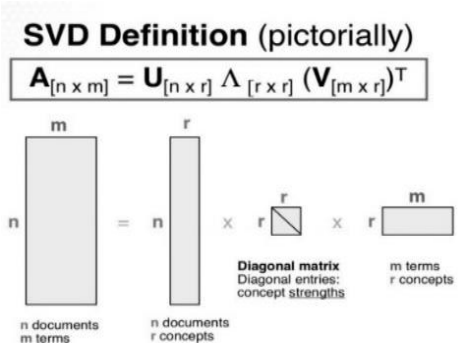
	brown	dog	fox	lazy	quick	red	slow	the	yellow
"the quick brown fox"	1	0	1	0	1	0	0	1	0
"the slow brown dog"	1	1	0	0	0	0	1	1	0
"the quick red fox"	0	1	0	0	1	1	0	1	0
"the lazy yellow fox"	0	0	1	1	0	0	0	1	1

Step 2: Singular Value Decomposition

The Singular Value Decomposition (SVD) of a matrix is a factorization of that matrix into three matrices. It has some interesting algebraic properties and conveys important geometrical and theoretical insights about linear transformations. It also has some important applications in data science. Interpretation of SVD is given by: Deriving mapping between m-dimensional space and r-dimensional singular vector space (rank of input matrix = r). Breaks down the original document into r linearly-independent base vectors or concepts. SVD can semantically cluster words and sentences by finding salient and vectors. The sentence that best represents this pattern will have the largest index value. After you run SVD and get the most salient concepts in the text, we need to select the sentences as summary.

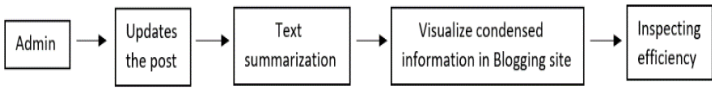
Step 1: Document term matrix

A document-term matrix is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms. This matrix is a specific instance of a document-feature matrix where "features" may refer to other properties of a document besides terms. They are useful in the field of natural language processing and computational text analysis.

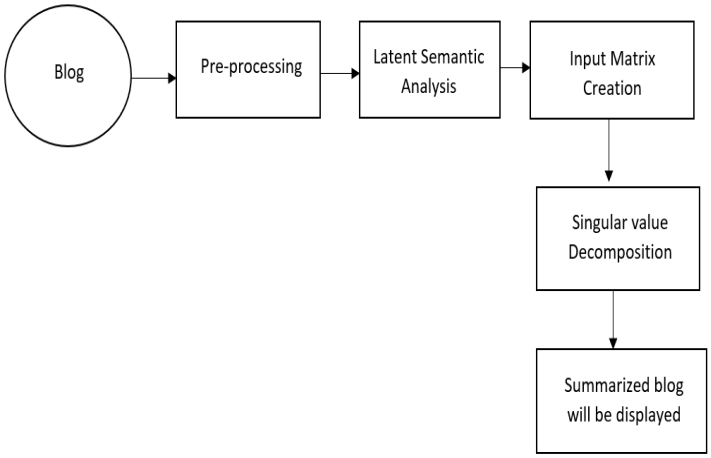


5. System Design – Architecture

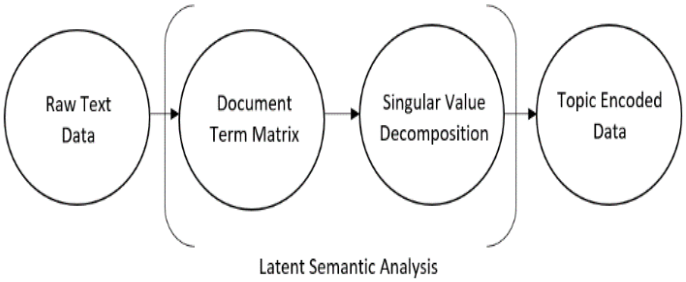
System View



Text Summarizer



Latent Semantic Analysis

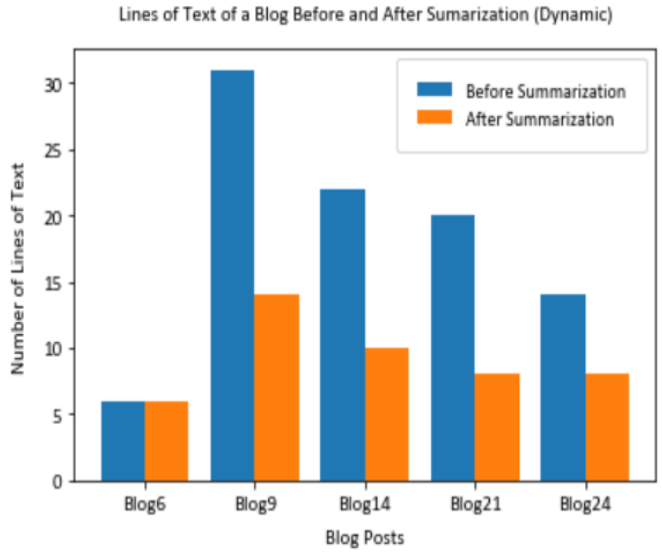


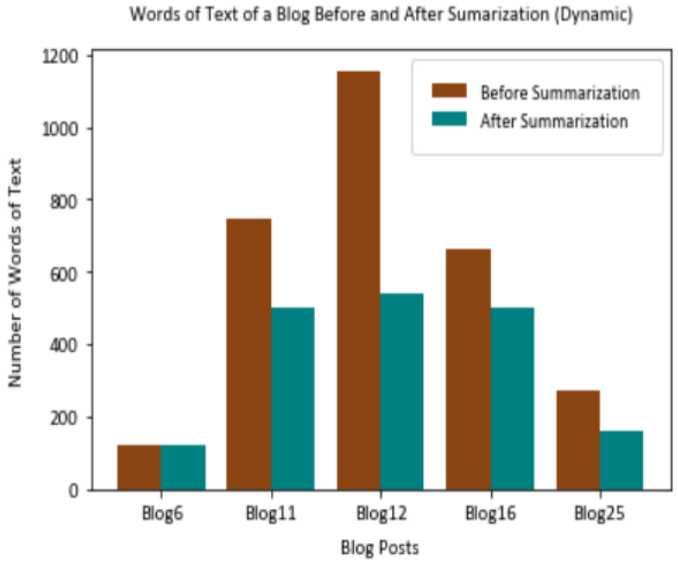
Description –

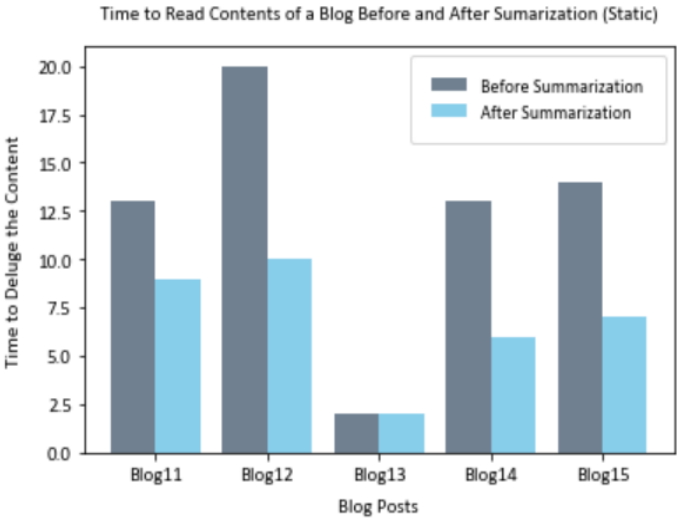
The blog mining involves admin who updates new blog, here by updated content is summarized in short and is stored into the database. Thus, summarized content is here by accessed by users while access of info regard a blog. This a part where the condensed information in blogging site is visualized. The next part is to inspect efficiency by gather of details regard info before summarization and after summarization. Thus, the pre-processed data is allowed to provide it for a graphical representation having a visual impact for comparison of before summarization and after summarization. During this process the tasks involved for text summarizer is blog content is pre-processed applying the algo Latent Semantic Analysis, thus displaying out summarized content of blog. The latent semantic analysis involves typically two steps, first step is document term matrix where the content gathered is represented in a matrix, and then followed by singular value decompose allows to extract the ranked text from the matrix and extracted data which is in encoded matrix format is extracted and pulled out as sentences. Here Argumentation mining occupies a position between natural language processing, argumentation theory and information retrieval. Argumentation mining aims to automatically detect, classify and structure argumentation in text. Argumentation mining focuses on the detection of all the arguments in a text and their relationships with their preceding and following arguments. Argumentation mining does not analyze the validity of the argumentation or its correctness. The aim is to detect those pieces of text which seem to function as argumentative (from a linguistic and semantic point of view) and the relations between them, i.e., their structure. Emotion argumentation means to evaluate the consistency of emotions from a set of premises to its corresponding conclusion.

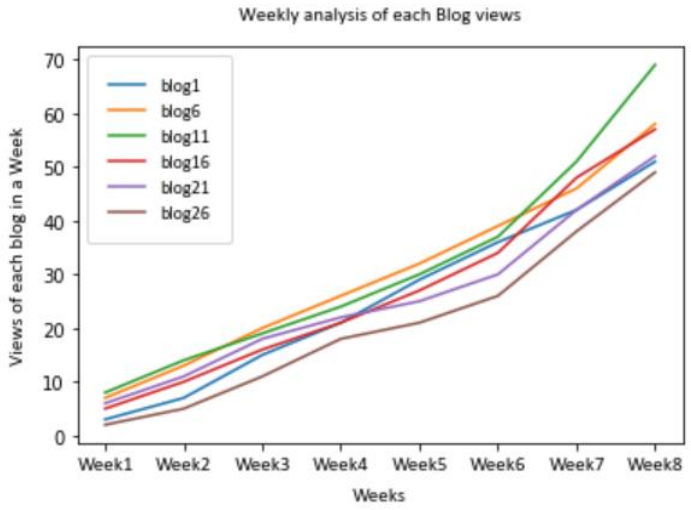
Results –

Analysis with readability, time and viewers –

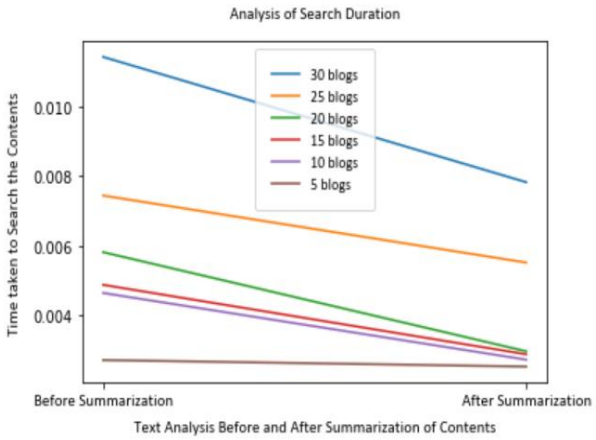


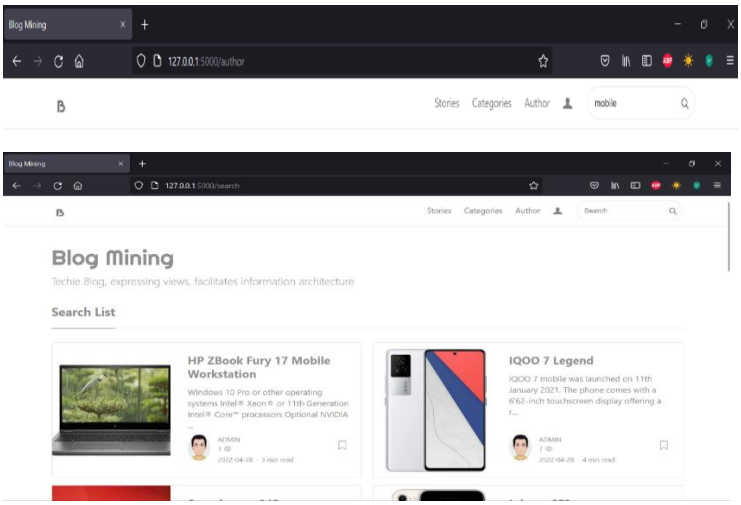




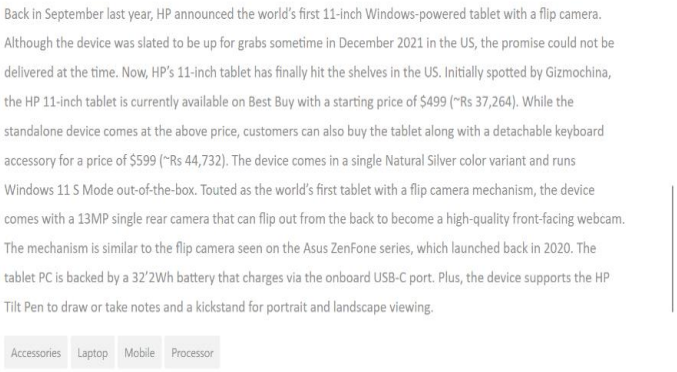


Search Analysis –

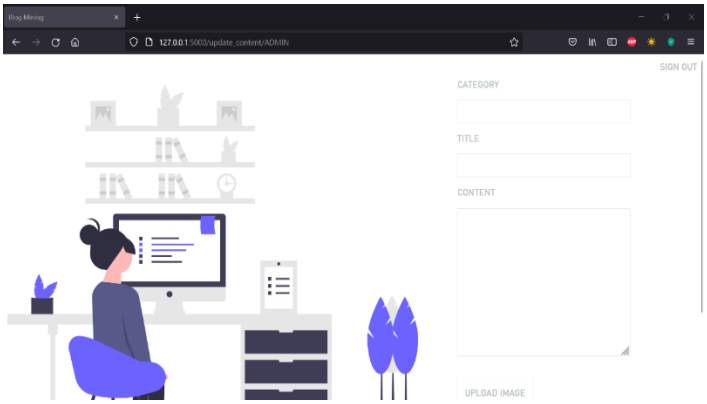




The summarized text view in blog –



Updation of summarized content by blogger –



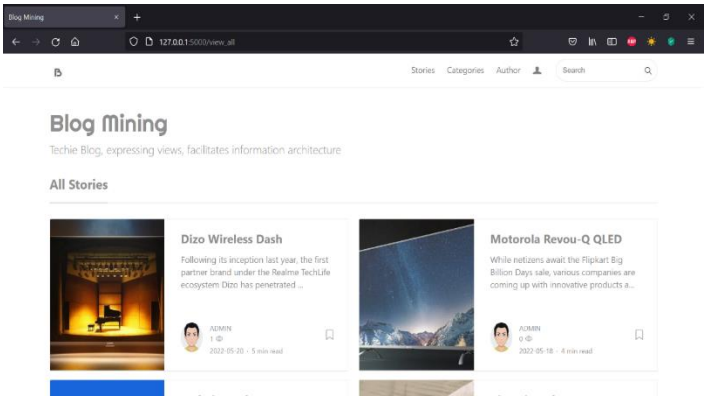
Conclusion –

The study of blog mining and emotion argumentation is used to analyze and search the online blog posts relevant contents in a quite simpler fashion. The main purpose includes provision of summarize content. As the information reflected is summarized view one can grasp away huge content in shorter duration and also browsing gives faster pace results. Blog mining can be concluded as a view by pulling out search optimization and also allows users to acknowledge deluge information of blogging websites.

Future Enhancements –

In future we can implement with various languages like Hindi. This allows more enrichment for users who prefer native language, the source having first language allows many users to put interest to look through vivid contents of the blogs.

The blog home page –



References –

[1] Thomas K. Landauer, Danielle S. McNamara, Simon Dennis, Walter Kintsch, Handbook of Latent Semantic Analysis
[2] Erkan and D. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization.
[3] Kazantseva and S. Szpakowicz. 2006. Challenges in evaluating summaries of short stories.
[4] D'Avanzo and B. Magnini. 2005. A keyphrase-based approach to summarization: The Lake system at DUC 2005. In *Proceedings of the Document Understanding Conference (DUC 2005)*.
[5] Mehwish Aziz, Muhammad Rafi, Sentence-Based Semantic Similarity measure for Blog Posts, 6th International Conference on Digital Content, Multimedia Technology and its Applications, IDC 2010, Aug 2010, IEEE Seoul.
[6] Alterman R., "Summarization in the small" in N. Sharkey edition - Advances in cognitive science. Chichester, England, Ellis Horwood. 1986.
[7] Alterman R., "Text summarisation" in Artificial Intelligence Review. 1990.
[8] Baxendale, P.B., "Man-made index for technical literature: an experiment", IBM Journal of Research and Development, 2, 4, 1958.
[9] Aho, A., Chang, S.-F., McKeown K,Radev, D., Smith,J., and Zaman, K. 1997. "Columbia digital news system: An environment for briefing and search over multimedia information". In Proceedings of IEEE ADL, Washington, DC.
[10] Edmundson H.P., "New methods in automatic extracting" in Journal of the ACM1, 6, 264-285. 1969.
[11] Cohen, J.D., "Highlights: Language- and Domain-Independent Automatic Indexing Terms for Abstracting", Journal of the American Society for Information Science.
[12] Garner, R., Efficient text summarization: costs and benefits, Journal of Educational Research.

AUTHORS

Ms. K. Vineela[1], Assistant Professor, Department of IT.
Sruthi Mandalapu[2] B.Tech, Vasireddy Venkatadri Institute of Technology
Sravya Mannava[3] B.Tech, Vasireddy Venkatadri Institute of Technology
Saraswathi Koduru[4] B.Tech, Vasireddy Venkatadri Institute of Technology
Veena Madhuri Konakanchi[5] B.Tech, Vasireddy Venkatadri Institute of Technology.