Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

PROJECT REPORT ON
"MAKING A WEBSITE FOR TEXT SUMMARIZATION"

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

School of Computer Science and Engineering, VIT Chennai

April 2024

Submitted by:

Rishu Singh (21BCE1306)

Chinmay Chaudhari (21BCE1250)

Ashwin Parathasarathy (21BCE1641)

Submitted to:

Dr. Lekshmi K

ACKNOWLEDGMENT

It gives us immense pleasure to present the report of the undertaken during the Bachelor of Technology 6th semester. We would like to express our deep and sincere gratitude to our project guide Dr. Lekshmi K for providing us with excellent guidance, encouragement, and inspiration throughout the mini-project work. Her logical thinking and understanding of "WEB DEVELOPMENT" have greatly influenced us in our project work. It was a great experience working with her because of her friendly nature. We are highly thankful to her for her immense support in all stages.

We would like to give special thanks to Dr. Nithyanandam P, Head of Department, SCOPE, for his immense support in all kinds of work and all the support we needed.

Name: Rishu Ravi Singh
Roll No.: 21BCE1306

Name: Chinmay Sanjay Chaudhari
Roll No.: 21BCE1250

Name: Ashwin Parathasarathy
Roll No.: 21BCE1641

DECLARATION

We hereby declare that this is our own work and that to the best of our knowledge and belief. It contains no material previously written or published by any other person nor material which to a substantial extent has been accepted for the award of a degree or other higher learning except where the acknowledgment has been made in the text.

Name: Rishu Ravi Singh
Roll No.: 21BCE1306

Name: Chinmay Sanjay Chaudhari
Roll No.: 21BCE1250

Name: Ashwin Parathasarathy
Roll No.: 21BCE1641

ABSTRACT

This paper presents a web-based application designed to facilitate efficient text summarization through the integration of multiple APIs. The platform is built using HTML, CSS, and Flask, incorporating an SQL database for user management, including secure login and signup functionalities. Users can easily navigate to a text input interface after authentication, where they are given the option to select from three different summarization APIs via a dropdown menu. This selection allows for customized summarization based on user preference and the specific requirements of the text involved. The application aims to enhance user engagement and productivity by providing quick, tailored summaries of extensive texts, making it an ideal tool for individuals seeking to optimize their reading and comprehension processes. The development and implementation of this text summarization tool demonstrate the potential for combining modern web technologies and API integration to create powerful, user-centric applications.

TABLE OF CONTENTS

INTRODUCTION

We present an innovative text summarization platform, designed to streamline your reading experience and enhance information absorption. This website leverages advanced technology, combining HTML, CSS, and Flask, to create a user-friendly interface that simplifies complex texts. Users can sign up or log in through our secure landing page, which connects to a robust SQL database for authentication and user management.

Once logged in, you're directed to a dedicated page where you can input any lengthy text. Our platform offers the unique feature of summarizing text using one of three different APIs, selectable via a dropdown menu. This flexibility allows users to choose the summarization style that best suits their needs, whether for academic, professional, or personal purposes.

Our goal is to provide a seamless and customizable experience that helps you save time and focus on what's important.

BUILDING OF OUR WEBSITE

The making of our website needs some elementary things as given below:

- HTML
- CSS
- VISUAL STUDIO CODE TO MAKE OUR WEBSITE
- GOOGLE CHROME TO RUN OUR WEBSITE

INTRO ABOUT WEBSITE

Our team has developed a sophisticated text summarization website designed to streamline the process of condensing large volumes of text into concise, manageable summaries. The site leverages HTML, CSS, and Flask, providing a user-friendly interface that caters to a variety of users, including professionals, students, and anyone in need of quick insights from lengthy documents or articles.

User Interface and Experience:

The journey on our site begins with a sleek, intuitive login/signup page crafted using HTML. This entry point requires users to authenticate by entering their email and password, linked to a securely hosted SQL database. The visual and interactive elements of the page are enhanced with CSS, ensuring that the forms are not only functional but also aesthetically appealing. Our use of responsive design principles guarantees that the website offers a consistent and engaging experience across all devices and screen sizes.

Server-Side Functionality:

Upon successful authentication, users are directed to the core feature of our website: the text input page. Our backend, powered by the Flask framework, handles all server-side logic including session management, user authentication, and database interactions. Flask also crucially manages the routing between the various pages and the integration with different text summarization APIs. Here, users are provided with a dropdown menu to select from one of three APIs for their text summarization needs, allowing for flexibility in choosing the method that best suits their specific requirements.

API Integration and Output Display:

Our website's ability to integrate multiple summarization APIs offers users a customizable experience. They can select the summarization service that best fits their needs based on style, tone, or output length. After the user's text is submitted and processed through the selected API, Flask retrieves the summarized content and delivers it back to the user. The results are dynamically displayed using HTML for structuring and CSS for styling, ensuring that the summarized text is both accessible and easy to read.

INTRODUCTION ABOUT THE LANGUAGES

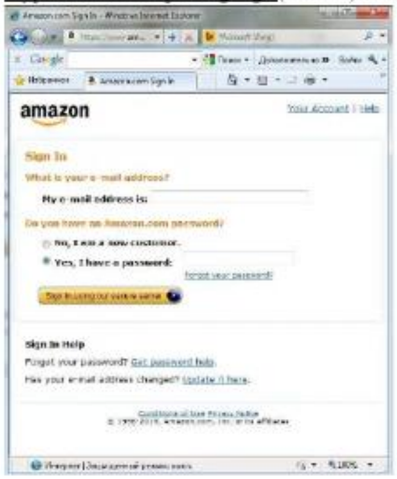
Three layers of web design: - HTML (Hyper Text Markup Language)

CSS (Cascading Style sheet)

JAVASCRIPT

HTML LANGUAGE

Hypertext Markup Language (HTML)is the set of markup symbols codes inserted into a file



amazon

Your Account | Help

Sign In

What is your e-mail address?

My e-mail address is:

Do you have an Amazon.com password?

No, I am a new customer.

Yes, I have a password:

Forgot your password?

Sign in using our secure server

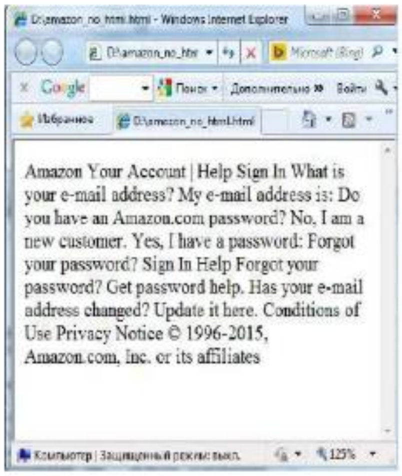
Sign In Help

Forgot your password? Get password help.

Has your e-mail address changed? Update it here.

Conditions of Use Privacy Notice

© 1996-2015, Amazon.com, Inc. or its affiliates

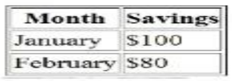


Amazon Your Account | Help Sign In What is your e-mail address? My e-mail address is: Do you have an Amazon.com password? No, I am a new customer. Yes, I have a password: Forgot your password? Sign In Help Forgot your password? Get password help. Has your e-mail address changed? Update it here. Conditions of Use Privacy Notice © 1996-2015, Amazon.com, Inc. or its affiliates

or

intended for display on the Internet. The markup tells web browsers how to display a web page's words and images.

WITH HTML

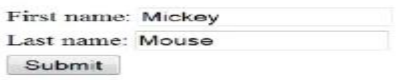


Month	Savings
January	$100
February	$80

WITHOUT HTML

Without any markup to give your page content structure, the browser renders **unformatted** and **disorganisation** text

HTML provides the structure **structure** and meaning to the to the website. **HTML tags** give content.



First name: Mickey

Last name: Mouse

Submit

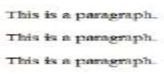
Some of the **HTML tags: -**

<p></p> - to organize text into paragraphs

<table></table> - to display table

<form></form> - to define form for user input

<img></img> - to add image



This is a paragraph.

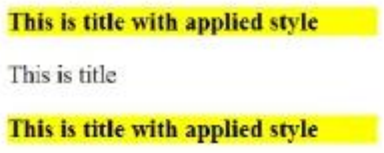
This is a paragraph.

This is a paragraph.

CSS LANGUAGE

CSS stands for Cascading Style Sheets. **CSS** describes how HTML elements are to be displayed on screen.

Some of the CSS demo are given below: -


This is title with applied style

This is title

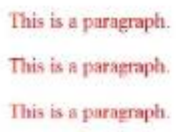
This is title with applied style

```
p {
color: red;
}
```

```
#title {
font-style: italic;
border: 1px dotted blue;
}
```

```
.title {
font-weight: bold;
background: yellow;
}
```

Things we can change with CSS: -


This is a paragraph.

This is a paragraph.

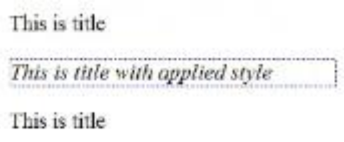
This is a paragraph.

Colours

Font

Font size

Backgrounds


This is title

This is title with applied style

This is title

Spacing sizes

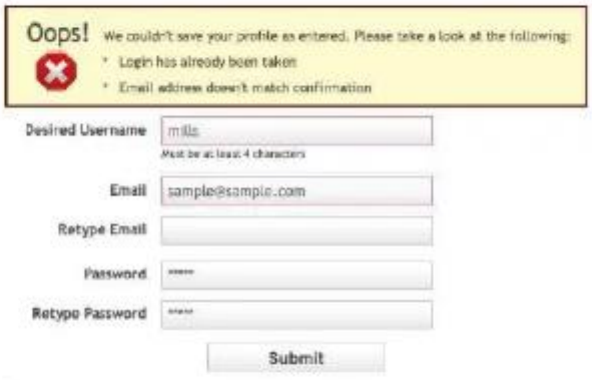
Borders

Positions (layout)

Things we can't change with CSS: -

Content

Markup


Oops! We couldn't save your profile as entered. Please take a look at the following:

- Login has already been taken
- Email address doesn't match confirmation

Desired Username	mills Must be at least 4 characters
Email	sample@sample.com
Retype Email	
Password	*****
Retype Password	*****

JAVASCRIPT LANGUAGE

JavaScript is a text-based programming language used both on the client-side and **server**-side that allows you to make web pages interactive. Where HTML and **CSS** are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user.

Given below are the things we can do with JavaScript: -

Luxoft Poland via Luxoft
May 6 at 4:15pm

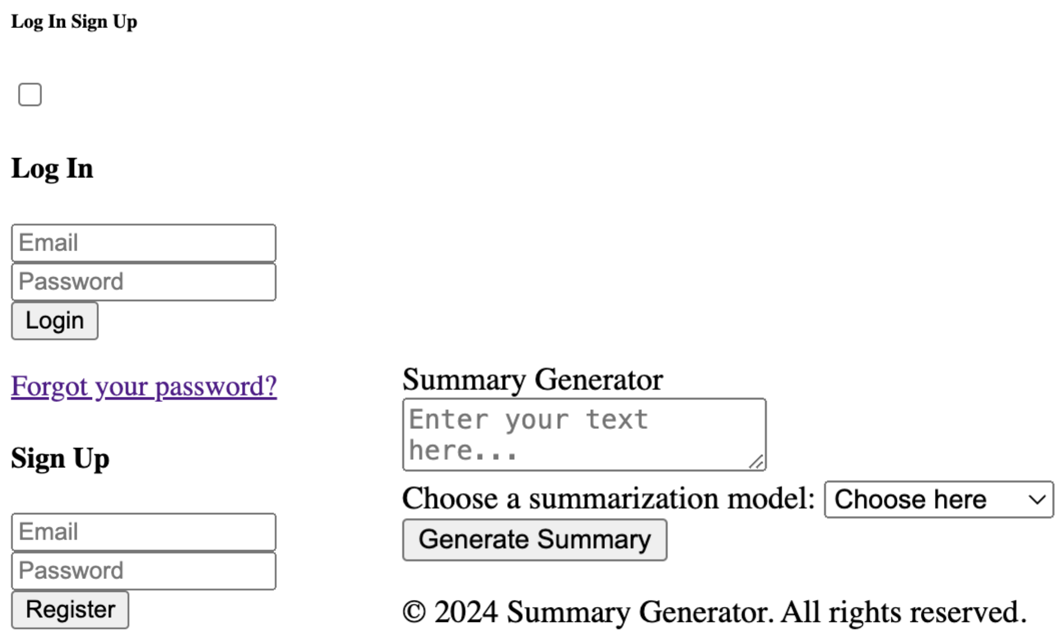
Luxoft rocks @ Agile Days 2015

Luxoft Rocks @ Agile Days 2015

Luxoft Team shined at Agile Days 2015 conference in Moscow, Russia. With its sleek booth, cool offerings and engaging speakers, Luxoft proved its expertise o...

YOUTUBE.COM

Validate values entered in the form fields.



Log In Sign Up

☐

Log In

Email

Password

Login

Forgot your password?

Sign Up

Email

Password

Register

Summary Generator

Enter your text here...

Choose a summarization model: Choose here

Generate Summary

© 2024 Summary Generator. All rights reserved.

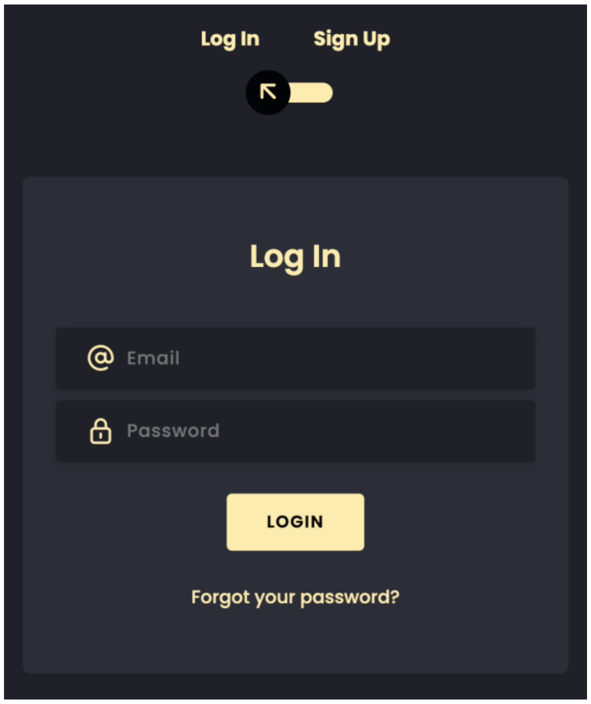
HTML CODE

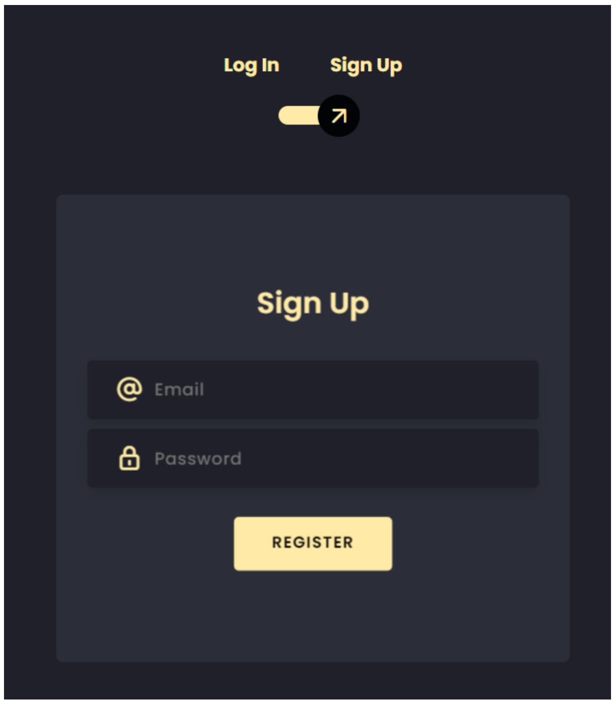
Main Page

```
<!doctype html>
<html lang="en">

<head>
   <title>Login Page</title>
   <meta charset="utf-8">
   <meta  name="viewport"  content="width=device-width,  initial-scale=1,  shrink-to-
fit=no">
   <link                                                                   rel="stylesheet"
href="https://unicons.iconscout.com/release/v2.1.9/css/unicons.css">
   <link    rel="stylesheet"    href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.5.0/css/bootstrap.min.css">
   <link rel="stylesheet" href="{{ url_for('static',filename='styles/style_login.css')
}}">
</head>

<body>
   <div class="section">
       <div class="container">
           <div class="row full-height justify-content-center">
               <div class="col-12 text-center align-self-center py-5">
                   <div class="section pb-5 pt-5 pt-sm-2 text-center">
                       <h6   class="mb-0   pb-3"><span>Log   In   </span><span>Sign
Up</span></h6>
                       <input class="checkbox" type="checkbox" id="reg-log" name="reg-
log" />
                       <label for="reg-log"></label>
                       <div class="card-3d-wrap mx-auto">
                           <div class="card-3d-wrapper">
                               <div class="card-front">
                                   <div class="center-wrap">
                                       <div class="section text-center">
                                           <h4 class="mb-4 pb-3">Log In</h4>
                                           <!-- Add this inside the body of your HTML
where you want the messages to appear -->
{% with messages = get_flashed_messages() %}
{% if messages %}
 {% for message in messages %}
```

```
   <div class="alert alert-success" role="alert">
     {{ message }}
   </div>
 {% endfor %}
{% endif %}
{% endwith %}

                                           <form action="login" method="post">
                                               <div class="form-group">
                                                   <input   type="email"   class="form-
style" placeholder="Email"

                                                       name="email">
                                                   <i   class="input-icon   uil   uil-
at"></i>

                                               </div>
                                               <div class="form-group mt-2">
                                                   <input type="password" class="form-
style" placeholder="Password"

                                                       name='password'>
                                                   <i  class="input-icon  uil  uil-lock-
alt"></i>

                                               </div>
                                               <button  name="Login-Button"  class="btn
mt-4" type="submit">Login</button>
                                           </form>
                                           <p    class="mb-0    mt-4    text-center"><a
href="https://www.web-leb.com/code"
                                                   class="link">Forgot            your
password?</a></p>
                                       </div>
                                   </div>
                               </div>
                               <div class="card-back">
                                   <div class="center-wrap">
                                       <div class="section text-center">
                                           <h4 class="mb-3 pb-3">Sign Up</h4>
                                           <form action="register" method="post">
                                               <div class="form-group mt-2">
                                                   <input   type="email"   class="form-
style" placeholder="Email" name="email" required>
                                                   <i   class="input-icon   uil   uil-
at"></i>
```

```
                                                </div>
                                                <div class="form-group mt-2">
                                                    <input type="password" class="form-
style" placeholder="Password" name="password" required>
                                                    <i class="input-icon uil uil-lock-
alt"></i>
                                                </div>
                                                <button           name="Register-Button"
class="btn mt-4" type="submit">Register</button>
                                            </form>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <script>
        document.addEventListener('DOMContentLoaded', function() {
            {% with messages = get_flashed_messages() %}
                {% if messages %}
                    Swal.fire({
                        title: 'Success!',
                        text: '{{ messages[0] }}',
                        icon: 'success',
                        confirmButtonText: 'Cool'
                    });
                {% endif %}
            {% endwith %}
        });
    </script>

</body>

</html>
```

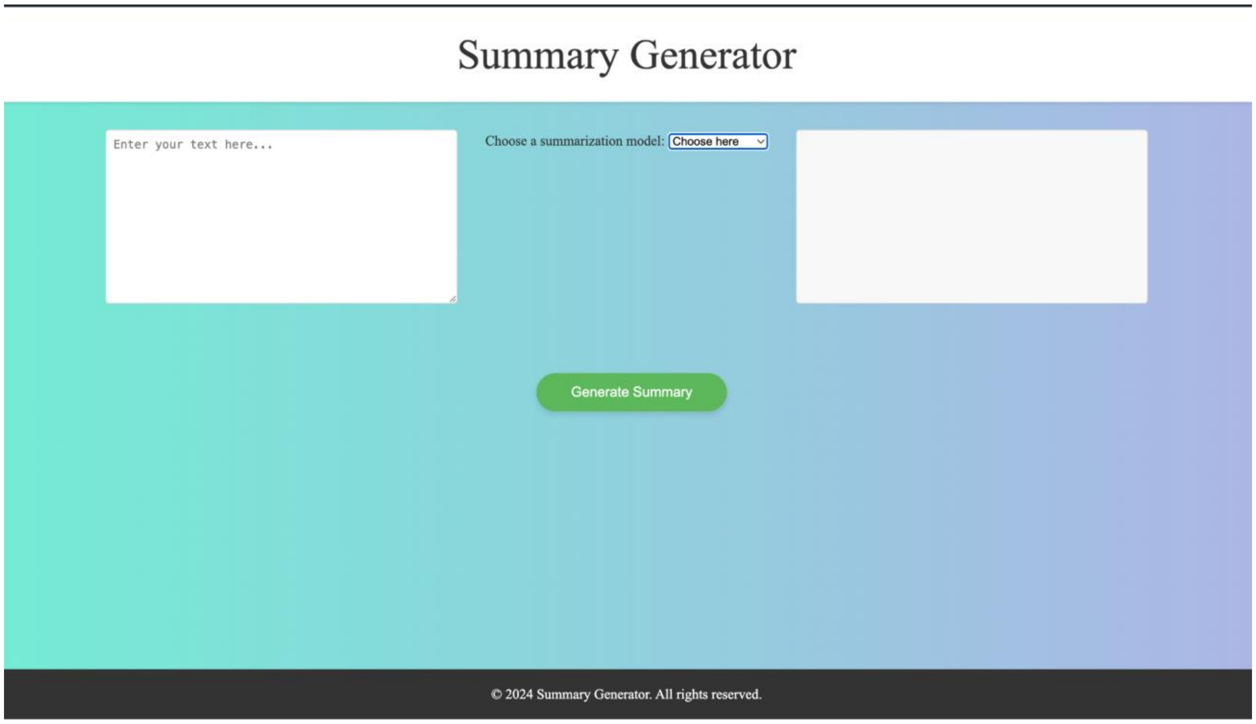

Log In
Sign Up
Log In
Email
Password
LOGIN
Forgot your password?


Log In
Sign Up
Sign Up
Email
Password
REGISTER

Summary Page

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Summary Generator</title>
<link rel= "stylesheet" type= "text/css" href= "{{
url_for('static',filename='styles/style_index.css') }}">
</head>
<body>
<header>
 <nav>
   <div class="logo">Summary Generator</div>
 </nav>
</header>

<main>
 <form action="content" method="post">
   <section id="generator">
     <div class="container">
       <textarea id="input-text" placeholder="Enter your text here..."
name="text"></textarea>

       <div>
         <label for="model-select">Choose a summarization model:</label>
         <select id="model-select" name="model">
           <option value="" selected disabled hidden>Choose here</option>
           <option value="azure">Microsoft Azure</option>
           <option value="mb">Meta Bart</option>
           <option value="fa">Falcon AI</option>
         </select>
       </div>

       <div id="summary-result">{{ summary_text | safe }}</div>
     </div>
     <button id="generate-btn" type="submit">Generate Summary</button>
   </section>
 </form>
</main>

<footer>
```

```
 <p>© 2024 Summary Generator. All rights reserved.</p>
</footer>
</body>
</html>
```

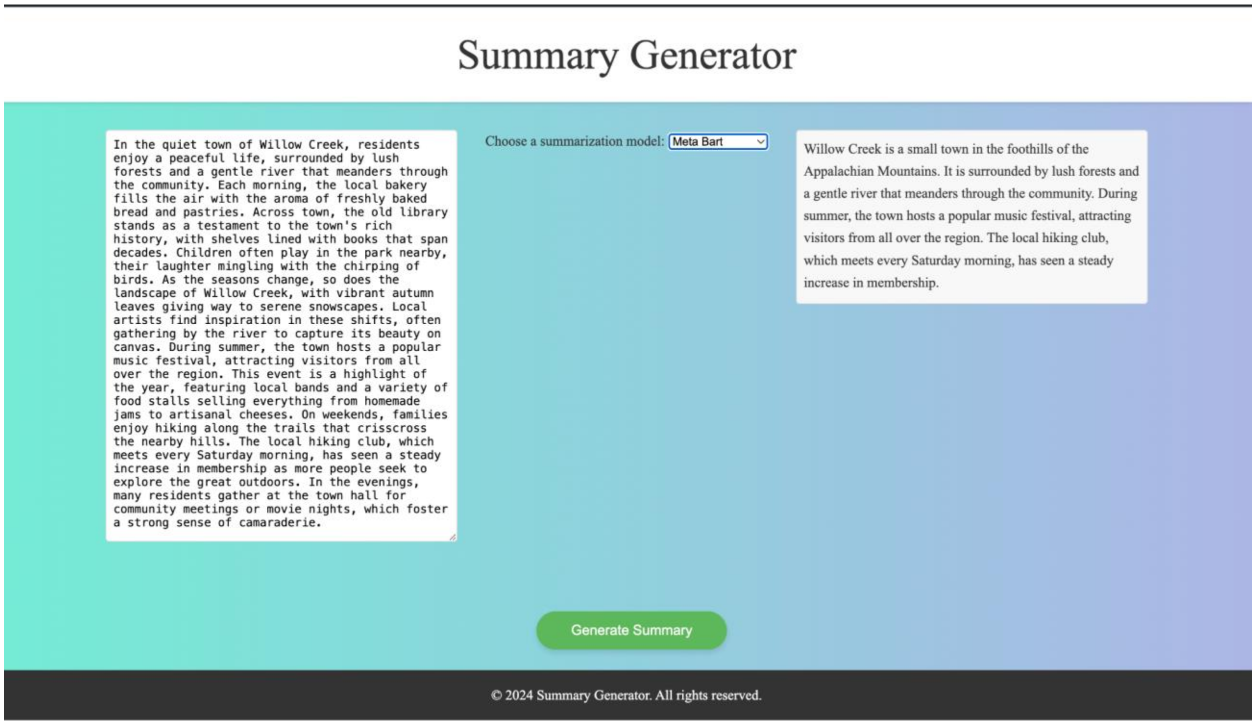


Summary Generator

Enter your text here...

Choose a summarization model: Choose here

© 2024 Summary Generator. All rights reserved.



Summary Generator

In the quiet town of Willow Creek, residents enjoy a peaceful life, surrounded by lush forests and a gentle river that meanders through the community. Each morning, the local bakery fills the air with the aroma of freshly baked bread and pastries. Across town, the old library stands as a testament to the town's rich history, with shelves lined with books that span decades. Children often play in the park nearby, their laughter mingling with the chirping of birds. As the seasons change, so does the landscape of Willow Creek, with vibrant autumn leaves giving way to serene snowscapes. Local artists find inspiration in these shifts, often gathering by the river to capture its beauty on canvas. During summer, the town hosts a popular music festival, attracting visitors from all over the region. This event is a highlight of the year, featuring local bands and a variety of food stalls selling everything from homemade jams to artisanal cheeses. On weekends, families enjoy hiking along the trails that crisscross the nearby hills. The local hiking club, which meets every Saturday morning, has seen a steady increase in membership as more people seek to explore the great outdoors. In the evenings, many residents gather at the town hall for community meetings or movie nights, which foster a strong sense of camaraderie.

Choose a summarization model: Meta Bart

Willow Creek is a small town in the foothills of the Appalachian Mountains. It is surrounded by lush forests and a gentle river that meanders through the community. During summer, the town hosts a popular music festival, attracting visitors from all over the region. The local hiking club, which meets every Saturday morning, has seen a steady increase in membership.

© 2024 Summary Generator. All rights reserved.

CSS CODE

Main Page

```
@import
url('https://fonts.googleapis.com/css?family=Poppins:400,500,600,700,800,900');
body{
 font-family: 'Poppins', sans-serif;
 font-weight: 300;
 line-height: 1.7;
 color: #ffeba7;
 background-color: #1f2029;
}
a:hover {
 text-decoration: none;
}
.link {
 color: #ffeba7;
}
.link:hover {
 color: #c4c3ca;
}
p {
 font-weight: 500;
 font-size: 14px;
}
h4 {
 font-weight: 600;
}
h6 span{
 padding: 0 20px;
 font-weight: 700;
}
.section{
 position: relative;
 width: 100%;
 display: block;
}
.full-height{
 min-height: 100vh;
}
[type="checkbox"]:checked,
```

```css
[type="checkbox"]:not(:checked){
display: none;
}
.checkbox:checked + label,
.checkbox:not(:checked) + label{
 position: relative;
 display: block;
 text-align: center;
 width: 60px;
 height: 16px;
 border-radius: 8px;
 padding: 0;
 margin: 10px auto;
 cursor: pointer;
 background-color: #ffeba7;
}
.checkbox:checked + label:before,
.checkbox:not(:checked) + label:before{
 position: absolute;
 display: block;
 width: 36px;
 height: 36px;
 border-radius: 50%;
 color: #ffeba7;
 background-color: #020305;
 font-family: 'unicons';
 content: '\eb4f';
 z-index: 20;
 top: -10px;
 left: -10px;
 line-height: 36px;
 text-align: center;
 font-size: 24px;
 transition: all 0.5s ease;
}
.checkbox:checked + label:before {
 transform: translateX(44px) rotate(-270deg);
}
.card-3d-wrap {
 position: relative;
 width: 440px;
 max-width: 100%;
 height: 400px;
 -webkit-transform-style: preserve-3d;
 transform-style: preserve-3d;
 perspective: 800px;
 margin-top: 60px;
}
.card-3d-wrapper {
 width: 100%;
 height: 100%;
 position:absolute;
 -webkit-transform-style: preserve-3d;
 transform-style: preserve-3d;
 transition: all 600ms ease-out;
}
.card-front, .card-back {
 width: 100%;
 height: 100%;
 background-color: #2b2e38;
 position: absolute;
 border-radius: 6px;
 -webkit-transform-style: preserve-3d;
}
.card-back {
 transform: rotateY(180deg);
}
.checkbox:checked ~ .card-3d-wrap .card-3d-wrapper {
 transform: rotateY(180deg);
}
.center-wrap{
 position: absolute;
 width: 100%;
 padding: 0 35px;
 top: 50%;
 left: 0;
 transform: translate3d(0, -50%, 35px) perspective(100px);
 z-index: 20;
 display: block;
}
.form-group{
 position: relative;
 display: block;
   margin: 0;
   padding: 0;
}
.form-style {
 padding: 13px 20px;
 padding-left: 55px;
 height: 48px;
 width: 100%;
 font-weight: 500;
 border-radius: 4px;
 font-size: 14px;
 line-height: 22px;
 letter-spacing: 0.5px;
 outline: none;
 color: #c4c3ca;
 background-color: #1f2029;
 border: none;
 -webkit-transition: all 200ms linear;
 transition: all 200ms linear;
 box-shadow: 0 4px 8px 0 rgba(21,21,21,.2);
}
.form-style:focus,
.form-style:active {
 border: none;
 outline: none;
 box-shadow: 0 4px 8px 0 rgba(21,21,21,.2);
}
.input-icon {
 position: absolute;
 top: 0;
 left: 18px;
 height: 48px;
 font-size: 24px;
 line-height: 48px;
 text-align: left;
 -webkit-transition: all 200ms linear;
  transition: all 200ms linear;
}
.btn{
 border-radius: 4px;
 height: 44px;
 font-size: 13px;
 font-weight: 600;
 text-transform: uppercase;
 -webkit-transition : all 200ms linear;
 transition: all 200ms linear;
 padding: 0 30px;
 letter-spacing: 1px;
 display: -webkit-inline-flex;
 display: -ms-inline-flexbox;
 display: inline-flex;
 align-items: center;
 background-color: #ffeba7;
 color: #000000;
}
.btn:hover{
 background-color: #000000;
 color: #ffeba7;
 box-shadow: 0 8px 24px 0 rgba(16,39,112,.2);
}
```

Summary Page

```
/* Basic Reset */
* {
 margin: 0;
 padding: 0;
 box-sizing: border-box;
}

body {
 font-family: 'Times New Roman', Tahoma, Geneva, Verdana, sans-serif;
 line-height: 1.6;
 background: linear-gradient(to right, #74ebd5, #acb6e5);
 color: #333;
 font-size: 100%;
}

header {
 background: #fff;
 padding: 1rem 0;
 box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
```

```css
nav .logo {
 text-align: center;
 font-size: 3rem;
 color: #333;

}

main {
 padding: 2rem;
 display: flex;
 justify-content: center;
 align-items: flex-start;
}

#generator .container {
 display: flex;
 width: 1200px;
 margin: 0 auto;
 gap: 2rem;
}

#input-text,
#summary-result {
 flex: 1;
 padding: 0.5rem;
 border: 1px solid #ddd;
 border-radius: 4px;
 height: 200px;
 transition: box-shadow 0.3s ease-in-out, border-color 0.3s ease-in-out;
 outline: none;
}

#input-text:focus
{
 border-color: #2d56dc; /* Change the border color to match the button or any color
you like */
 box-shadow: 0 0 8px 2px #2d56dc;
}

#summary-result {
 background: #f8f8f8;
}
```

```css
#generate-btn {
 padding: 0.5rem 2rem;
 background-color: #5cb85c;
 color: #fff;
 border: none;
 border-radius: 4px;
 cursor: pointer;
 transition: background-color 0.3s ease;
 margin-top: 1rem; /* Aligns with the top of textarea */
}


#generate-btn {
 padding: 0.8rem 2.5rem;
 background-color: #5cb85c; /* Primary button color */
 color: white;
 font-size: 1rem;
 border: none;
 border-radius: 25px; /* Rounded corners */
 cursor: pointer;
 outline: none;
 position: relative; /* For pseudo-elements */
 overflow: hidden; /* Ensures pseudo-elements don't overflow */
 box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1); /* Subtle shadow for depth */
 transition: all 1s ease; /* Smooth transition for hover effects */
 margin-left: 31rem;
 margin-top: 5rem;
}

#generate-btn::before {
 content: '';
 position: absolute;
 top: 50%;
 left: 50%;
 transform: translate(-50%, -50%) scale(0);
 width: 300%;
 height: 300%;
 background-color: rgba(255, 255, 255, 0.2); /* Light overlay for ripple effect */
 border-radius: 50%;
 transition: transform 0.5s, opacity 0.3s ease;
 z-index: 0;
 opacity: 0;
}

#generate-btn:hover::before {
 transform: translate(-50%, -50%) scale(1);
 opacity: 1;
}

#generate-btn:hover {
 background-color: #4cae4c; /* Slightly darker shade on hover */
}

#generate-btn:focus {
 box-shadow: 0 0 0 2px #5cb85c; /* Focus outline for accessibility */
}

#generate-btn span {
 position: relative; /* Ensure the text stays above the pseudo-elements */
 z-index: 1;
 }



footer {
 background: #333;
 color: #fff;
 text-align: center;
 padding: 1rem 0;
 position: absolute;
 bottom: 0;
 width: 100%;
}
```

FLASK CODE

App.py

```
from flask import Flask, render_template, redirect, url_for, request, flash
import mysql.connector
import os
import requests
from mysql.connector import Error

app = Flask(__name__)
app.secret_key = '1234'

key = os.environ.get('LANGUAGE_KEY')
endpoint = os.environ.get('LANGUAGE_ENDPOINT')
print(key, endpoint, end="/n")

from azure.ai.textanalytics import TextAnalyticsClient
from azure.core.credentials import AzureKeyCredential

# Authenticate the client using your key and endpoint
def authenticate_client():
   ta_credential = AzureKeyCredential(str(key))
   text_analytics_client = TextAnalyticsClient(
           endpoint=str(endpoint),
           credential=ta_credential)
   return text_analytics_client

client = authenticate_client()

def create_database_connection():
   connection = None
   try:
       connection = mysql.connector.connect(
           host='sql6.freemysqlhosting.net',
           database='sql6698831',
           user='sql6698831',
           password='GxkHsHh8eu'
       )
       print("MySQL Database connection successful")
   except Error as e:
       print(f"The error '{e}' occurred")
   return connection
```

```
def create_database(connection, query):
   cursor = connection.cursor()
   try:
       cursor.execute(query)
       print("Database created successfully")
   except Error as e:
       print(f"The error '{e}' occurred")

def create_table(connection, query):
   cursor = connection.cursor()
   try:
       cursor.execute(query)
       print("Table created successfully")
   except Error as e:
       print(f"The error '{e}' occurred")

@app.route('/')
def home():
   return render_template('main.html')

@app.route('/login', methods=['POST'])
def login():
   # Here you would add logic to validate login credentials
   # For now, we'll just redirect to the main page
   email = request.form['email']
   password = request.form['password']  # In a real app, use hashed passwords
   if email=="admin@gmail.com":
       return render_template('index.html')
   conn = create_database_connection()
   create_users_table = """
CREATE TABLE IF NOT EXISTS users (
   id INT AUTO_INCREMENT PRIMARY KEY,
   email VARCHAR(255) UNIQUE NOT NULL,
   password VARCHAR(255) NOT NULL
);
"""

   create_table(conn, create_users_table)
   cursor = conn.cursor()
   try:
       cursor.execute("SELECT password FROM users WHERE email = %s", (email,))
```

```
        record = cursor.fetchone()
        if record:
            stored_password = record[0]
            if password == stored_password:  # Replace this line with hashed password
check in production
                # User is authenticated
                return render_template('index.html')  # Redirect to a home or dashboard
page
            else:
                # Wrong password
                flash('Wrong password. Please try again!')
                return redirect(url_for('main'))
        else:
            # Email does not exist
            flash('User not found')
            return redirect(url_for('main'))
    except Error as e:
        flash(f'An error occurred: {str(e)}', 'error')
        return redirect(url_for('main'))
    finally:
        cursor.close()
        conn.close()

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']  # Consider hashing the password for
security
        conn = create_database_connection()
        cursor = conn.cursor()
        try:
            cursor.execute("INSERT INTO users (email, password) VALUES (%s, %s)",
(email, password))
            conn.commit()
            flash('User registered successfully!')
            return redirect(url_for('main'))  # Redirect to clear form
        except Error as e:
            flash('Error: ' + str(e), 'error')
        finally:
            cursor.close()
            conn.close()
```

```python
   return render_template('main.html')

def sample_extractive_summarization(client, text):
   from azure.core.credentials import AzureKeyCredential
   from azure.ai.textanalytics import (
       TextAnalyticsClient,
       ExtractiveSummaryAction
   )

   text+=" "
   document = [text]
   poller = client.begin_analyze_actions(
       document,
       actions=[
           ExtractiveSummaryAction(max_sentence_count=4)
       ],
   )

   document_results = poller.result()
   for result in document_results:
       extract_summary_result = result[0]  # first document, first result
       res = " ".join([sentence.text for sentence in
extract_summary_result.sentences])
   return res

@app.route('/content', methods=['POST'])
def handle_content():
   text = request.form['text']  # Match the name attribute of your textarea
   model = request.form['model']
   if model=="azure":
       #res = sample_extractive_summarization(client, text)
       a = 1

   elif model=="mb":
       API_URL = "https://api-inference.huggingface.co/models/facebook/bart-large-cnn"
       headers = {"Authorization": "Bearer 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"}
       payload = {
                   "inputs": text,
                   "options":{
                       "wait_for_model": "true"
                   }
                   }
```

```
        response = requests.post(API_URL, headers=headers, json=payload)
        res = response.json()

    elif model=="fa":
        API_URL = "https://api-
inference.huggingface.co/models/Falconsai/text_summarization"
        headers = {"Authorization": "Bearer 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"}
        payload = {
                    "inputs": text,
                    "options":{
                        "wait_for_model": "true"
                    }
                    }
        response = requests.post(API_URL, headers=headers, json=payload)
        res = response.json()

    return render_template('index.html', summary_text = res[0].get('summary_text'))

@app.route('/main')
def main():
    # Render the main page
    return render_template('index.html')

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

CONCLUSION

In conclusion, our team's website stands as a robust tool that combines the technical capabilities of HTML, CSS, and Flask to deliver a dynamic and efficient text summarization service. This platform not only boosts user productivity but also provides flexible options to accommodate a wide range of summarization preferences. Through thoughtful design and powerful technology, we aim to transform how users interact with and consume large texts.