Home.html <head>

<link rel=”preconnect” href=<https://fonts.googleapis.com>>

<link rel=”preconnect” href=<https://fonts.gstatic.com> crossorigin>

<link href=<https://fonts.googleapis.com/css2?family=Pacifico&display=swap> rel=”stylesheet”>

<style> body{

Display: grid; place-content:center; background-image:

url(‘https://www.parents.com/thmb/FmihFthdF62Ou0EoUlRI7Ikeb4k=/1500x0/filters:n o\_upscale():max\_bytes(150000):strip\_icc()/GettyImages-884378360-2000-

6d8b0cd7ce0c4daf96ed15c80e3bfc16.jpg’); background-size: 100%; background-position: center;

Font-family: ‘Lucida Sans’, ‘Lucida Sans Regular’, ‘Lucida Grande’, ‘Lucida

Sans Unicode’, Geneva, Verdana, sans-serif; }

.container{ display:grid; place-content: center; padding:100px; background: linear-gradient(to right, rgb(228, 228, 101),rgb(97, 200, 216)); border-radius:30px;

}

#enter{ width:500px; height: 40px; border-radius:20px; padding-left: 20px; padding-right:20px; border:none;

}

#btn{ color:black; border: 2px solid red; background-color: transparent; height: 40px; border-radius:10px; transition:0.25s ease-in;

}

#btn:hover{ background-color: red; transition:0.25s ease-in; color:white; cursor:pointer;

}

#result{ max-width: 400px; height: auto; padding:20px; background-color: rgb(48, 48, 48); color:white; border-radius: 20px;

}

H2{ text-align: center; font-size: 40px; font-family: “Pacifico”, cursive; background: linear-gradient(to right, red, rgb(99, 69, 232));

-webkit-background-clip: text; background-clip: text; color: transparent;

}

</style>

</head>

<body>

<div class=”container”>

<h2>SentiVerse</h2>

<div id=”response”>

<form>

<center><input id=”enter” placeholder=”Enter your statement” name=”length” type=”text”></center>

<br>

<br>

<center><input type=”submit” id=”btn” onclick=”document.getElementById(‘result’).style.display = block” value=”Analyse mental state”></center>

</form>

</div>

<br>

<center>

<div id=”result”>

{{password}}

</div>

</center>

<br>

<center><button id=”btn” class=”speak”>Speak</button></center>

<br>

<form action=”{% url ‘password’ %}”>

<center><button id=”btn” class=”speak”>Get today’s task</button></center>

</form>

<br>

<form action=”{% url ‘about’ %}”>

<center><button id=”btn” class=”speak”>Get report</button></center>

</form>

</div>

<script src=<https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js> integrity=”sha384-

geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68Siy3Te4Bkz”

crossorigin=”anonymous”></script>

<script>

// Check if browser supports speech synthesis if (‘speechSynthesis’ in window) {

Const speakButton = document.querySelector(‘.speak’); const content = document.getElementById(‘result’);

// Event listener for button click

speakButton.addEventListener(‘click’, () => { // Create a new speech synthesis object const speechSynth = window.speechSynthesis; // Get the text content of the div const textToRead = content.textContent;

// Create a new SpeechSynthesisUtterance object const utterance = new SpeechSynthesisUtterance(textToRead);

Const voices = speechSynthesis.getVoices();

Const ziraVoice = voices.find(voice => voice.name === ‘Microsoft David Desktop – English (United States)’);

// Set the Zira voice utterance.voice = ziraVoice;

// Function to speak after a delay function speakAfterDelay() {

// Speak the text speechSynth.speak(utterance);

}

// Speak the text speechSynth.speak(utterance);

});

} else { alert(‘Your browser does not support speech synthesis.’);

}

</script>

</body>

About.html

<head>

<link rel=”preconnect” href=<https://fonts.googleapis.com>>

<link rel=”preconnect” href=<https://fonts.gstatic.com> crossorigin>

<link href=<https://fonts.googleapis.com/css2?family=Pacifico&display=swap> rel=”stylesheet”>

<style> body{

Display: grid; place-content:center; background-image:

url(‘https://www.parents.com/thmb/FmihFthdF62Ou0EoUlRI7Ikeb4k=/1500x0/filters:n o\_upscale():max\_bytes(150000):strip\_icc()/GettyImages-884378360-2000-

6d8b0cd7ce0c4daf96ed15c80e3bfc16.jpg’); background-size: 100%; background-position: center;

Font-family: ‘Lucida Sans’, ‘Lucida Sans Regular’, ‘Lucida Grande’, ‘Lucida

Sans Unicode’, Geneva, Verdana, sans-serif; }

.container{ display:grid; place-content: center; padding:100px; background: linear-gradient(to right, rgb(228, 228, 101),rgb(97, 200, 216)); border-radius:30px;

}

#enter{

Width:500px; height: 40px; border-radius:20px; padding-left: 20px; padding-right:20px; border:none;

}

#btn{ color:black; border: 2px solid red; background-color: transparent; height: 40px; border-radius:10px; transition:0.25s ease-in;

}

#btn:hover{ background-color: red; transition:0.25s ease-in; color:white; cursor:pointer;

}

#result{ max-width: 400px; text-wrap: wrap; height: auto; padding:20px; background-color: rgb(48, 48, 48); color:white; border-radius: 20px;

}

H2{ text-align: center; font-size: 40px; font-family: “Pacifico”, cursive; background: linear-gradient(to right, red, rgb(99, 69, 232));

-webkit-background-clip: text; background-clip: text; color: transparent;

}

</style>

</head>

<body>

<div class=”container”>

<h2>SentiVerse</h2>

<div id=”response”>

<form>

<center><h2>Overall report on your past sentiments</h2></center>

<center>

<div id=”result”>

{{password}}

</div>

</center>

<br>

<center><button id=”btn” class=”speak”>Speak</button></center>

</div>

<script src=<https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js> integrity=”sha384-

geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68Siy3Te4Bkz”

crossorigin=”anonymous”></script>

<script>

// Check if browser supports speech synthesis if (‘speechSynthesis’ in window) {

Const speakButton = document.querySelector(‘.speak’); const content = document.getElementById(‘result’);

// Event listener for button click

speakButton.addEventListener(‘click’, () => { // Create a new speech synthesis object const speechSynth = window.speechSynthesis; // Get the text content of the div const textToRead = content.textContent;

// Create a new SpeechSynthesisUtterance object const utterance = new SpeechSynthesisUtterance(textToRead);

Const voices = speechSynthesis.getVoices();

Const ziraVoice = voices.find(voice => voice.name === ‘Microsoft David Desktop – English (United States)’);

// Set the Zira voice utterance.voice = ziraVoice;

// Function to speak after a delay function speakAfterDelay() {

// Speak the text speechSynth.speak(utterance);

}

// Speak the text speechSynth.speak(utterance);

});

} else { alert(‘Your browser does not support speech synthesis.’);

}

</script>

</body>

Feedback.html

<head>

<link rel=”preconnect” href=<https://fonts.googleapis.com>>

<link rel=”preconnect” href=<https://fonts.gstatic.com> crossorigin>

<link href=<https://fonts.googleapis.com/css2?family=Pacifico&display=swap> rel=”stylesheet”>

<style> body{

Display: grid; place-content:center; background-image:

url(‘https://www.parents.com/thmb/FmihFthdF62Ou0EoUlRI7Ikeb4k=/1500x0/filters:n o\_upscale():max\_bytes(150000):strip\_icc()/GettyImages-884378360-2000-

6d8b0cd7ce0c4daf96ed15c80e3bfc16.jpg’); background-size: 100%; background-position: center;

Font-family: ‘Lucida Sans’, ‘Lucida Sans Regular’, ‘Lucida Grande’, ‘Lucida

Sans Unicode’, Geneva, Verdana, sans-serif; }

.container{ display:grid; place-content: center; padding:100px; background: linear-gradient(to right, rgb(228, 228, 101),rgb(97, 200, 216)); border-radius:30px;

}

#enter{ width:500px; height: 40px; border-radius:20px; padding-left: 20px; padding-right:20px; border:none;

}

#btn{ color:black; border: 2px solid red; background-color: transparent; height: 40px; border-radius:10px; transition:0.25s ease-in;

}

#btn:hover{ background-color: red; transition:0.25s ease-in; color:white; cursor:pointer;

}

#result{ max-width: 400px; text-wrap: wrap; height: auto; padding:20px; background-color: rgb(48, 48, 48); color:white; border-radius: 20px;

}

H2{ text-align: center; font-size: 40px; font-family: “Pacifico”, cursive; background: linear-gradient(to right, red, rgb(99, 69, 232));

-webkit-background-clip: text; background-clip: text; color: transparent;

}

</style>

</head>

<body>

<div class=”container”>

<h2>SentiVerse</h2>

<div id=”response”>

<form>

<center><h2>TODAY’S TASK</h2></center>

<center>

<div id=”result”>

{{password}}

</div>

</center>

<br>

<center><button id=”btn” class=”speak”>Speak</button></center>

</div>

<script src=<https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js> integrity=”sha384-

geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68Siy3Te4Bkz”

crossorigin=”anonymous”></script>

<script>

// Check if browser supports speech synthesis if (‘speechSynthesis’ in window) {

Const speakButton = document.querySelector(‘.speak’); const content = document.getElementById(‘result’);

// Event listener for button click

speakButton.addEventListener(‘click’, () => { // Create a new speech synthesis object const speechSynth = window.speechSynthesis; // Get the text content of the div const textToRead = content.textContent;

// Create a new SpeechSynthesisUtterance object const utterance = new SpeechSynthesisUtterance(textToRead);

Const voices = speechSynthesis.getVoices();

Const ziraVoice = voices.find(voice => voice.name === ‘Microsoft David Desktop – English (United States)’);

// Set the Zira voice utterance.voice = ziraVoice;

// Function to speak after a delay function speakAfterDelay() {

// Speak the text speechSynth.speak(utterance);

}

// Speak the text speechSynth.speak(utterance);

});

} else { alert(‘Your browser does not support speech synthesis.’);

}

</script>

</body>

Views.py

From 48jango.shortcuts import render from 48jango.http import HttpResponse import random; import openai import os import textwrap import google.generativeai as genai

GOOGLE\_API\_KEY=’’ genai.configure(api\_key=GOOGLE\_API\_KEY) model = genai.GenerativeModel(‘gemini-pro’)

X = []

Sentiment = “”

Def to\_markdown(text):

Text = text.replace(‘•’, ‘ \*’) return textwrap.indent(text, ‘’, predicate=lambda \_: True)

Def home(request):

Global x

Length=request.GET.get(‘length’) instructions = f””” You are given with a role of a psychiatrist. You will be provided

With a sentence delimited within triple backticks or an paragraph and you should analyse the sentiment of the paragraph and tell the emotions of the writer of the paragraph.you should only specify the exact emotion of the person whether they are happy,sad or having mixed emotions like you seem to sad. If it is sad or mixed recommend some suggestions which are tasks to improve their mental state or if it is happy show some uplifting message like “You are stronger than you think, and you have the power to overcome any challenge that comes your way.”

“Believe in yourself, trust your journey, and keep moving forward one step at a time.”

“You are capable of achieving great things. Keep working hard and stay focused on your goals.” Like that but not these alone along with it.

-the result should be within 50 words and the suggestion should be a task like

Mindfulness meditation ,Physical activity,Social support, Self-care, Gratitude practice, Limiting negative influences, Setting realistic goals, Deep breathing exercises but not these things alone.

* If the sentence is about any context other than sentiment analysis, then you should return Iam designed for sentiment analysis only. This is very important. For example if the sentence is about coding a software, then tell Iam designed for sentiment analysis only.

-sentence = ```{length}```

-You should not assist them in any other context other than sentiment analysis. You should say Iam designed for sentiment analysis only if the context is not about sentiment analysis.

-you should not use the word writer instead you may address them as you “””

# thepassword = get\_completion(instructions)

Response = model.generate\_content(instructions) result = to\_markdown(response.text)

Chatbot\_response = result sentiment = result

x.append(chatbot\_response) return render(request, ‘generator/home.html’, {‘password’: chatbot\_response })

def password(request):

global x global sentiment

length=request.GET.get(‘length’) instructions = f””” You are given with a role of a psychiatrist. You will be provided

with a sentence delimited within triple backticks or an paragraph and you should analyse the sentiment of the paragraph and tell the emotions of the person who said the paragraph.

* If the sentence is about any context other than sentiment analysis, then you should return I don’t know. This is very important. For example if the sentence is about coding a software, then tell I don’t know.

-sentence = ```{length}```

-You should not assist them in any other context other than sentiment analysis. You should say I don’t know if the context is not about sentiment analysis.

“””

# thepassword = get\_completion(instructions)

Response = model.generate\_content(instructions) result = to\_markdown(response.text)

Chatbot\_response = result sentiment = result

x.append(chatbot\_response)

return render(request, ‘generator/home.html’, {‘password’: chatbot\_response })

def give\_feedback(request):

global x global sentiment

instructions = f””” You are given with a role of a psychiatrist.

You have to generate an task and should give that as today’s task to the user to improve their mental state by doing that task for that day alone and should not not repeat the task.

-paragraph = ```{sentiment}```

-please don’t give the array as the response.

-do not use bold texts.

-Use only 51jango51. 51jango51e lines with line breaks. This is important.

Give a single paragraph answer within 100 words. This is very very important.

“”” response = model.generate\_content(instructions) result = to\_markdown(response.text) chatbot\_response = result

Return render(request, ‘generator/feedback.html’, {‘password’: chatbot\_response })

Def generate\_report(request):

Global x global sentiment

Instructions = f””” You are given with a role of a psychiatrist. You will be provided with a array delimited within triple backticks. The array consists of the past sentiment analysis of the user.

You have to give the overall report to the user based on the past sentiments in the array. Consider the array is sorted that is first element in the array is the first sentiment, and the last is the latest sentiment.

-paragraph = ```{x}```

-please don’t give the array as the response.

-do not use bold texts.

-Use only 52jango52. 52jango52e lines with line breaks. This is important.

Give a single paragraph answer within 250 words. This is very very important.

“””

Response = model.generate\_content(instructions) result = to\_markdown(response.text) chatbot\_response = result return render(request, ‘generator/about.html’, {‘password’: chatbot\_response })

Def about(request):

Return render(request,’generator/about.html’)

# def get\_completion(prompt, model=”gpt-3.5-turbo”): # Andrew mentioned that the prompt/ completion paradigm is preferable for this class

# messages = [{“role”: “user”, “content”: prompt}]

# response = openai.ChatCompletion.create(

# model=model,

# messages=messages,

# temperature=0, # this is the degree of randomness of the model’s output

# )

# return response.choices[0].message[“content”]

Settings.py

“””

Django settings for password\_generator project.

Generated by ‘django-admin startproject’ using Django 4.2.1.

For more information on this file, see <https://docs.djangoproject.com/en/4.2/topics/settings/>

For the full list of settings and their values, see <https://docs.djangoproject.com/en/4.2/ref/settings/> “””

From pathlib import Path

# Build paths inside the project like this: BASE\_DIR / ‘subdir’.

BASE\_DIR = Path( file ).resolve().parent.parent

# Quick-start development settings – unsuitable for production

# See <https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/>

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = ‘django-insecure- c5t+wqtyi#&ox!%lqy2y=wr@7ee25vcofkw&2k5#lbry5b-%u=’

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

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

‘django.contrib.admin’,

‘django.contrib.auth’, ‘django.contrib.contenttypes’,

‘django.contrib.sessions’,

‘django.contrib.messages’,

‘django.contrib.staticfiles’,

‘generator’,

]

MIDDLEWARE = [

‘django.middleware.security.SecurityMiddleware’,

‘django.contrib.sessions.middleware.SessionMiddleware’,

‘django.middleware.common.CommonMiddleware’,

‘django.middleware.csrf.CsrfViewMiddleware’,

‘django.contrib.auth.middleware.AuthenticationMiddleware’,

‘django.contrib.messages.middleware.MessageMiddleware’,

‘django.middleware.clickjacking.XframeOptionsMiddleware’,

]

ROOT\_URLCONF = ‘password\_generator.urls’

TEMPLATES = [

{

‘BACKEND’: ‘django.template.backends.django.DjangoTemplates’,

‘DIRS’: [],

‘APP\_DIRS’: True,

‘OPTIONS’: {

‘context\_processors’: [

‘django.template.context\_processors.debug’,

‘django.template.context\_processors.request’,

‘django.contrib.auth.context\_processors.auth’,

‘django.contrib.messages.context\_processors.messages’,

],

},

},

]

WSGI\_APPLICATION = ‘password\_generator.wsgi.application’

# Database

# <https://docs.djangoproject.com/en/4.2/ref/settings/#databases>

DATABASES = {

‘default’: {

‘ENGINE’: ‘django.db.backends.sqlite3’,

‘NAME’: BASE\_DIR / ‘db.sqlite3’,

}

}

# Password validation

# <https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators>

AUTH\_PASSWORD\_VALIDATORS = [

{

‘NAME’:

‘django.contrib.auth.password\_validation.UserAttributeSimilarityValidator’,

},

{

‘NAME’: ‘django.contrib.auth.password\_validation.MinimumLengthValidator’,

},

{

‘NAME’: ‘django.contrib.auth.password\_validation.CommonPasswordValidator’,

},

{

‘NAME’: ‘django.contrib.auth.password\_validation.NumericPasswordValidator’,

},

]

# Internationalization

# <https://docs.djangoproject.com/en/4.2/topics/i18n/>

LANGUAGE\_CODE = ‘en-us’

TIME\_ZONE = ‘UTC’

USE\_I18N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# <https://docs.djangoproject.com/en/4.2/howto/static-files/>

STATIC\_URL = ‘static/’

# Default primary key field type

# <https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field> DEFAULT\_AUTO\_FIELD = ‘django.db.models.BigAutoField’

Urls.py

“””

URL configuration for password\_generator project.

The `urlpatterns` list routes URLs to views. For more information please see: <https://docs.djangoproject.com/en/4.2/topics/http/urls/> Examples:

Function views

Add an import: from my\_app import views

Add a URL to urlpatterns: path(‘’, views.home, name=’home’)

Class-based views

Add an import: from other\_app.views import Home

Add a URL to urlpatterns: path(‘’, Home.as\_view(), name=’home’)

Including another URLconf

Import the include() function: from 58jango.urls import include, path

Add a URL to urlpatterns: path(‘blog/’, include(‘blog.urls’))

“”” from 58jango.urls import path from generator import views urlpatterns = [ path(‘’,views.home,name=”home”),

Path(‘password/’,views.give\_feedback, name=”password”), path(‘about/’,views.generate\_report,name=”about”),

]