Back End - Database Development

Image Generator - Pinterest Version (Original Image Generator)

Code Blame 55 lines (46 loc) · 2.06 KB

```python
import hashlib
import os
from pinscrape import pinscrape

def hash_file(filepath):
    hasher = hashlib.sha256()
    with open(filepath, 'rb') as f:
        buf = f.read()
        hasher.update(buf)
    return hasher.hexdigest()

def remove_duplicates(directory):
    downloaded_hashes = set()
    for filename in os.listdir(directory):
        filepath = os.path.join(directory, filename)
        if os.path.isfile(filepath):
            file_hash = hash_file(filepath)
            if file_hash in downloaded_hashes:
                os.remove(filepath)
                print(f"Duplicate removed: {filename}")
            else:
                downloaded_hashes.add(file_hash)

def download_images(search_term, output_directory, proxy_list, num_threads, max_images):
    details = pinscrape.scraper.scrape(search_term, output_directory, proxy_list, num_threads, max_images)

    if details["isDownloaded"]:
        print(f"\nDownloading for '{search_term}' completed !!")
        print(f"\nTotal urls found for '{search_term}': {len(details['extracted_urls'])}")
        print(f"\nTotal images downloaded for '{search_term}': {len(details['url_list'])}")
        remove_duplicates(output_directory)
    else:
        print(f"\nNothing to download for '{search_term}' !!")

def is_deepest_folder(directory):
    # A folder is deepest if it contains no subdirectories
    return not any(os.path.isdir(os.path.join(directory, sub)) for sub in os.listdir(directory) if not sub.startswith('.'))

def get_deepest_folders(base_directory):
    deepest_folders = []
    for root, dirs, files in os.walk(base_directory):
        if is_deepest_folder(root):
            deepest_folders.append(root)
    return deepest_folders

# Base directory where the folders are located
base_directory = "INSERT DIRECTORY HERE"

# Get the deepest folders
deepest_folders = get_deepest_folders(base_directory)

# Loop through the deepest folders and download images using the folder name as the search term
for folder in deepest_folders:
    search_term = os.path.basename(folder)
    download_images(search_term, folder, {}, 10, 1000)
```

This Python script is designed to automate the process of scraping and downloading images from Pinterest. It utilizes search terms derived from the names of the deepest subdirectories within a specified base directory, ensuring a topic-specific collection of images. Additionally, the script includes functionality to remove duplicate images based on content hashing, maintaining a unique set of images in each directory.

Features

- Automatic Image Downloading: Downloads images from Pinterest based on directory-derived search terms.
- Duplicate Removal: Ensures all downloaded images are unique within their respective directories.
- Multi-threading Support: Allows for faster downloads by leveraging multiple threads.
- Proxy Support: Capable of using proxy lists to bypass network restrictions or for privacy reasons.

Image Generator - BulkAI Version (Updated Image Generator)

```python
import yaml
import os

# Ask for user input
album_name = input("What would you like to call the album? ")
prompt_base = input("What prompt do you want? (E.g., Photo realistic image of the front of a brick house) ")
num_images = int(input("How many images do you want? "))

# Generate prompts based on the number of images
prompts = [f"{prompt_base}-{i+1}" for i in range(num_images)]

# Configuration dictionary
config = {
    "bot": "midjourney",
    "album": album_name,
    "download": True,
    "upscale": True,
    "variation": True,  # Set based on your preference
    "thumbnail": False,
    "suffix": " --ar 3:2",
    "prompt": prompts,
    "wait": "180s"  # Adds a 20-second wait between prompts
}

# Define the YAML file path
yaml_file_path = os.path.expanduser("~/INSERT FILE PATH HERE/bulkai.yaml")

# Save the configuration to a YAML file
with open(yaml_file_path, "w") as file:
    yaml.dump(config, file, default_flow_style=False)

print("YAML configuration file has been generated at:", yaml_file_path)
```

The **Generate Images.py** script automates the process of generating images. Below is a detailed breakdown of its functionality:

Input Collection

- Album Name: Users are prompted to enter a name for the image album.
- Image Prompt: Users provide a base prompt which is used to define the theme or concept for the images.
- Number of Images: Users specify how many images they want to generate.

Prompt Construction

- The script constructs a list of unique prompts for each image by appending an index to the base prompt.

Configuration and Output

- Configuration: All user inputs and settings are compiled into a configuration dictionary.
- YAML Configuration: The configuration is saved to a YAML file, allowing for repeated use or modification.
- **Bot Compatibility:** The script is tailored to interact with the midjourney bot.
- Options:
    - Download: Enables downloading of generated images.

- Upscale: Allows for the enhancement of image resolution.
- Variation: Offers variation in image generation.
- Thumbnail: Option to generate thumbnails is available but disabled by default.
- Suffix: Custom suffixes can be added to commands as needed.
- Wait Time: Configurable wait time between issuing prompts to manage API calls or bot interactions.

The YAML configuration file location is set by default to ~INSERT FILE PATH/bulkai.yaml, and users are notified of the file path upon successful creation. The BulkAI.yaml file can be located below:

```yaml
album: /Users/opt/anaconda3/bin/python "/Users/INSERT FILE DIRECTORY
  Images/Generate Images.py"
bot: midjourney
download: true
prompt:
- Photo realistic image of the front of a brick house-1
- Photo realistic image of the front of a brick house-2
suffix: ' --ar 3:2'
thumbnail: false
upscale: true
variation: true
wait: 180s
```

Image Pre-Processing Script

This Python script is designed for pre-processing images for machine learning tasks, particularly for preparing datasets for training, testing, and validation in a Convolutional Neural Network (CNN) or other image-based machine learning models. The script includes functionalities such as resizing, augmenting, normalising, and adding noise to images, followed by splitting the dataset into training, testing, and validation sets.

Features

- Resizing: Changes the dimensions of the images to 1024x1024 pixels to ensure uniformity.
- Augmentation: Applies grayscale conversion, rotation, and horizontal flipping to images to enhance the robustness of the model.
- Normalisation: Adjusts pixel values to a range of [0, 1] for better model convergence.
- Noise Addition: Introduces random noise to images to reduce overfitting and improve generalization.
- Dataset Splitting: Divides the processed images into training, testing, and validation sets based on standard splitting ratios.

The code for this is below:

Code Blame 84 lines (71 loc) · 3.28 KB

```python
import os
import random
import numpy as np
from PIL import Image, ImageOps
from sklearn.model_selection import train_test_split

def resize_image(image, size=(1024, 1024)):
    """Resize the image to the specified size."""
    return image.resize(size, Image.LANCZOS)

def augment_image(image):
    """Apply augmentation techniques: grayscale conversion, rotation, and flipping."""
    # Convert to grayscale
    if random.choice([True, False]):
        image = ImageOps.grayscale(image)

    # Rotate the image by a random angle between -25 and 25 degrees
    if random.choice([True, False]):
        image = image.rotate(random.uniform(-25, 25))

    # Flip the image horizontally
    if random.choice([True, False]):
        image = ImageOps.mirror(image)

    return image

def normalise_image(np_image):
    """Normalize image pixel values to the range [0, 1]."""
    return np_image.astype('float32') / 255.0

def add_noise(np_image, noise_factor=0.05):
    """Add random noise to the image."""
    noise = np.random.randn(*np_image.shape) * noise_factor
    np_image_noisy = np_image + noise
    return np.clip(np_image_noisy, 0., 1.)

def process_images(image_dir):
    """Process all images in the specified directory."""
    print(f"Processing images in {image_dir}")
    processed_images = []
    for root, dirs, files in os.walk(image_dir):
        for filename in files:
            if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                with Image.open(os.path.join(root, filename)) as img:
                    img = resize_image(img)
                    img = augment_image(img)
                    np_img = np.array(img)
                    np_img = normalise_image(np_img)
                    np_img = add_noise(np_img)
                    processed_images.append((np_img, filename))
    print(f"Finished processing images in {image_dir}")
    return processed_images

def split_and_save_images(images, output_dir, folder_path):
    """Split images into train and test sets and save."""
    category = os.path.basename(folder_path)
    if not images:
        print(f"No images found for {category}")
        return
    print(f"Splitting and saving images for {category}")
    train, test = train_test_split(images, test_size=0.4, random_state=42)

    for dataset, name in [(train, 'train'), (test, 'test')]:
        folder_path = os.path.join(output_dir, name, category)
        if not os.path.exists(folder_path):
            os.makedirs(folder_path)
        for img, filename in dataset:
            Image.fromarray((img * 255).astype(np.uint8)).save(os.path.join(folder_path, filename))
    print(f"Finished splitting and saving images for {category}")

def process_folders(root_dir, processed_images_directory):
    """Process each subdirectory within the root directory."""
    for root, dirs, files in os.walk(root_dir):
        for dir in dirs:
            folder_path = os.path.join(root, dir)
            print(f"Processing and splitting folder: {folder_path}")
            images = process_images(folder_path)
            split_and_save_images(images, processed_images_directory, folder_path)

root_dir = r'C:\\Users\\INSERT DIRECTORY HERE'
processed_images_directory = r'C:\\Users\\INSERT DIRECTORY HERE'

# Call the function to start processing
process_folders(root_dir, processed_images_directory)
```

Back End - Model Development

Generator:

```
import tensorflow as tf
from keras import layers


def build_generator(noise_dim, output_channels=3, activation="tanh", alpha=0.2):
    inputs1 = layers.Input(shape=(noise_dim,))
    label_input = layers.Input(shape=(1,))

    label_embedding = layers.Embedding(5, 10)(label_input)
    label_embedding = layers.Flatten()(label_embedding)

    concatenated_input = layers.Concatenate()([inputs1, label_embedding])
    #concatenated_input = inputs1
    x = layers.Dense(512 * 4 * 4, use_bias=False)(concatenated_input)
    x = layers.Reshape((4, 4, 512))(x)

    x = layers.Conv2DTranspose(64 * 8, kernel_size=5, strides=2, padding='same')(x)
    x = layers.BatchNormalization()(x)
    x = layers.LeakyReLU(0.2)(x)

    x = layers.Conv2DTranspose(64 * 4, kernel_size=5, strides=2, padding='same')(x)
    x = layers.BatchNormalization()(x)
    x = layers.LeakyReLU(0.2)(x)

    x = layers.Conv2DTranspose(64 * 2, kernel_size=5, strides=2, padding='same')(x)
    x = layers.BatchNormalization()(x)
    x = layers.LeakyReLU(0.2)(x)

    x = layers.Conv2DTranspose(64 * 1, kernel_size=5, strides=2, padding='same')(x)
    x = layers.BatchNormalization()(x)
    x = layers.LeakyReLU(0.2)(x)
    x = layers.Dropout(0.3)(x)

    outputs = layers.Conv2D(3, kernel_size=5, padding='same', activation="tanh", dtype='float32')(x)

    model = tf.keras.Model(inputs=[inputs1, label_input], outputs=outputs, name='generator')
    return model
```

This python script works to build the generator component of our NN architecture, this means that when called by the system it creates a ‘model’ object with arbitrary weights which acts as the untrained version of the neural network responsible for converting random noise to images once the whole architecture is trained.

Input pre-processing and concatenation:

- At the beginning of the script multiple pieces of input are taken in tandem (the label and noise)
- The label is embedded and expanded to be the same size as the image.
- They are then concatenated in order to enter the network proper.

Primary learning loop:

- The core of the network consists of four rounds of transpositional convolution, which is a process by which the input noise is made larger, and learned features are applied to it in a black box pattern.
- This process is normalized by batch normalization, which prevents uncommon features being too prevalent.
- This process is aided by Elu layers which give the neural network an extra set of weights though which to nullify certain data points.

Output:

- After features of the output are randomly stripped to prevent overfitting, they are passed back to the training loop.

Discriminator:

```
def build_discriminator(img_shape=(64, 64, 3), activation='linear', alpha=0.2):
    con_label = layers.Input(shape=(1,))
    label_embedding = layers.Embedding(5, 10)(con_label) # was 30
    label_embedding = layers.Flatten()(label_embedding)
    label_embedding = layers.Dense(img_shape[0] * img_shape[1])(label_embedding)
    label_embedding = layers.Reshape((img_shape[0], img_shape[1], 1))(label_embedding)

    stream1_input = layers.Input(shape=(64, 64, 3))
    merge = layers.Concatenate()([stream1_input, label_embedding])
    #merge = stream1_input

    x = layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same', use_bias=False)(merge)
    #x = layers.LayerNormalization()(x)
    x = layers.LeakyReLU()(x)

    x = layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same', use_bias=False)(x)
    #x = layers.LayerNormalization()(x)
    x = layers.LeakyReLU()(x)

    x = layers.Conv2D(256, (5, 5), strides=(2, 2), padding='same', use_bias=False)(x)
    #x = layers.LayerNormalization()(x)
    x = layers.LeakyReLU()(x)

    x = layers.Conv2D(512, (5, 5), strides=(2, 2), padding='same', use_bias=False)(x)
    #x = layers.LayerNormalization()(x)
    x = layers.LeakyReLU()(x)

    x = layers.Flatten()(x)
    x = layers.Dropout(0.3)(x)
    x = layers.Dense(1, dtype='float32')(x)

    model = tf.keras.Model([stream1_input, con_label], x)
    return model
```

This script works to build the discriminator for the network architecture (initializing it in the same way as the generator build script), however this network has the function of learning the distribution of both the generator's output, and that of the database; and then trying to separate the two.

Input pre-processing:

- This network takes input in the same format as the generator, but with images instead of noise, and with both real and fake images.
- Here both image and noise are embedded and transformed (since the network reduces the size of the output)

Main learning loop:

- This network's core consists of four rounds of convolution, where previously recognised features in the image are extracted into distinct images containing only those features.
- This is stabilized and supported once again by batch normalization and LeakyRelu.

Output:

- The output is flattened before being dropped out, since it is otherwise multi-dimensional and cannot be passed to the vector function used to establish discriminator loss.

Back End - Training loop:

```
        gradients_norm = tf.sqrt(tf.reduce_sum(tf.square(gradients), axis=[1, 2, 3]))

        gradient_penalty = tf.reduce_mean((gradients_norm - 1.0) ** 2)

        return gradient_penalty

    def train_step(self, real_samples: tf.Tensor) -> typing.Dict[str, float]:
        real_samples, labels = real_samples
        batch_size = tf.shape(real_samples)[0]
        noise = tf.random.normal([batch_size, self.noise_dim])
        gps = []

        for _ in range(self.discriminator_extra_steps):

            with tf.GradientTape() as tape:
                fake_samples = self.generator([noise, labels], training=True)
                pred_real = self.discriminator([real_samples, labels], training=True)
                pred_fake = self.discriminator([fake_samples, labels], training=True)

                real_samples = self.add_instance_noise(real_samples)
                fake_samples = self.add_instance_noise(fake_samples)

                gp = self.gradient_penalty(real_samples, fake_samples, self.discriminator, labels)
                gps.append(gp)

                disc_loss = self.discriminator_loss(pred_real, pred_fake) + gp * self.gp_weight

            grads = tape.gradient(disc_loss, self.discriminator.trainable_variables)

            self.discriminator_opt.apply_gradients(zip(grads, self.discriminator.trainable_variables))

        with tf.GradientTape() as tape:
            fake_samples = self.generator([noise, labels], training=True)
            pred_fake = self.discriminator([fake_samples, labels], training=True)
            gen_loss = self.generator_loss(pred_fake)

        grads = tape.gradient(gen_loss, self.generator.trainable_variables)

        self.generator_opt.apply_gradients(zip(grads, self.generator.trainable_variables))

        self.compiled_metrics.update_state(real_samples, fake_samples)

        results = {m.name: m.result() for m in self.metrics}
        results.update({"d_loss": disc_loss, "g_loss": gen_loss, "gp": tf.reduce_mean(gps)})

        return results
```

This script works to undertake one epoch of training for the model, (this then being called several hundred times till the whole database has been iterated, and this in turn 100 times). It consists of a complex forward propagation function, involving the propagation of multiple models and their aggregation to a gradient for standard back propagation.

Discriminator propagation:

- Fake images are generated.
- Then the discriminator is tested on both real and fake images.

Gradient penalty propagation:

- Then noise is added to the real and fake data. *1
- Then the discriminator is forward propagated on the aggregation of real and fake.

Discriminator loss:

- Then the discriminator's loss is calculated as its ability to distinguish between the means of real and fake (from discriminator propagation), weighted by the gradient penalty.

Generator loss:

- Then (much more simply) the generator's loss is calculated using a heuristic for the inverse of the discriminator's ability.

*1

```
def gradient_penalty(
        self,
        real_samples: tf.Tensor,
        fake_samples: tf.Tensor,
        discriminator: tf.keras.models.Model,
        labels
    ) -> tf.Tensor:
    batch_size = tf.shape(real_samples)[0]

    epsilon = tf.random.uniform(shape=[batch_size, 1, 1, 1], minval=0, maxval=1)

    interpolated_samples = epsilon * real_samples + ((1 - epsilon) * fake_samples)

    with tf.GradientTape() as tape:
        tape.watch(interpolated_samples)
        logits = discriminator([interpolated_samples, labels], training=True)

    gradients = tape.gradient(logits, interpolated_samples)

    gradients_norm = tf.sqrt(tf.reduce_sum(tf.square(gradients), axis=[1, 2, 3]))

    gradient_penalty = tf.reduce_mean((gradients_norm - 1.0) ** 2)

    return gradient_penalty
```

```
import os
import cv2
import typing
import imageio
import numpy as np
import tensorflow as tf
from keras.callbacks import TensorBoard
from model import build_generator, build_discriminator


class WGAN_GP(tf.keras.models.Model):
    def __init__(
            self,
            discriminator: tf.keras.models.Model,
            generator: tf.keras.models.Model,
            noise_dim: int,
            discriminator_extra_steps: int=5,
            gp_weight: typing.Union[float, int]=10.0
        ) -> None:
        super(WGAN_GP, self).__init__()
        self.discriminator = discriminator
        self.generator = generator
        self.noise_dim = noise_dim
        self.discriminator_extra_steps = discriminator_extra_steps
        self.gp_weight = gp_weight

    def compile(
            self,
            discriminator_opt: tf.keras.optimizers.Optimizer,
            generator_opt: tf.keras.optimizers.Optimizer,
            discriminator_loss: typing.Callable,
            generator_loss: typing.Callable,
            **kwargs
        ) -> None:
        super(WGAN_GP, self).compile(**kwargs)
        self.discriminator_opt = discriminator_opt
        self.generator_opt = generator_opt
        self.discriminator_loss = discriminator_loss
        self.generator_loss = generator_loss

    def add_instance_noise(self, x: tf.Tensor, stddev: float = 0.1) -> tf.Tensor:
        noise = tf.random.normal(tf.shape(x), mean=0.0, stddev=stddev, dtype=x.dtype)
        return x + noise
```

This script serves to instantiate the various variables for the training loop, these variables can be split into 4 sections:

Model and library declaration:

- Here blank network objects are called from keras.
- Then certain local parameters (such as latent-dim) are defined for them.

Model instantiation:

- Here the models (as defined earlier) are called to replace the blank model objects, and updated with their learned weights.
- Furthermore the learned state of the gradient penalty is also called.

Optimiser declaration and compilation:

- Here the optimizers for the models are declared and then called (in the compilation subroutine) so that training can begin.

```
        return results


class ResultsCallback(tf.keras.callbacks.Callback):
    def __init__(
            self,
            noise_dim: int,
            output_path: str,
            examples_to_generate: int=5,
            grid_size: tuple=(5, 1),
            spacing: int=5,
            gif_size: tuple=(416, 416),
            duration: float=0.1,
            save_model: bool=True
        ) -> None:
        super(ResultsCallback, self).__init__()
        self.seed = tf.random.normal([examples_to_generate, noise_dim])
        self.results = []
        self.output_path = output_path
        self.results_path = output_path + '/results'
        self.grid_size = grid_size
        self.spacing = spacing
        self.gif_size = gif_size
        self.duration = duration
        self.save_model = save_model

        os.makedirs(self.results_path, exist_ok=True)

    def save_plt(self, epoch: int, results: np.ndarray):
        w, h , c = results[0].shape
        grid = np.zeros((self.grid_size[0] * w + (self.grid_size[0] - 1) * self.spacing, self.grid_size[1] * h + (self.grid_size[1] - 1) * self.spacing, c),
dtype=np.uint8)
        for i in range(self.grid_size[0]):
            for j in range(self.grid_size[1]):
                grid[i * (w + self.spacing):i * (w + self.spacing) + w, j * (h + self.spacing):j * (h + self.spacing) + h] = results[i * self.grid_size[1] + j]

        grid = cv2.cvtColor(grid, cv2.COLOR_RGB2BGR)

        cv2.imwrite(f'{self.results_path}/img_{epoch}.png', grid)

        self.results.append(cv2.resize(grid, self.gif_size, interpolation=cv2.INTER_AREA))

    def on_epoch_end(self, epoch: int, logs: dict=None):
        labels = tf.constant([1, 2, 3, 4, 5])
        predictions = self.model.generator([self.seed, labels], training=False)
        predictions_uint8 = (predictions * 127.5 + 127.5).numpy().astype(np.uint8)
        self.save_plt(epoch, predictions_uint8)

        if self.save_model:
            models_path = os.path.join(self.output_path, "model")
            os.makedirs(models_path, exist_ok=True)
            self.model.discriminator.save(models_path + "/discriminator.h5")
            self.model.generator.save(models_path + "/generator.h5")

    def on_train_end(self, logs: dict=None):
        imageio_images = [imageio.core.util.Image(image[...,::-1]) for image in self.results]
        imageio.mimsave(self.results_path + "/output.gif", imageio_images, duration=self.duration)
```

This script serves to define the saveable weights for the model, then update them at various points, then save them to be used in future epochs or iterations.

Instantiation:

- Defines the components necessary to save the weights of the model as parameters to a class.

- Then instantiates that class for the current epoch, so that its state can be saved independently.

Save data:

- This subroutine convolves weighting data passed to it into a flat vector.
- Then saves it to a CSV file for later retrieval.

On_epoch_end:

- This subroutine updates the model weights after each loop of train_step, based on the information calculated during it.
- This information is stored (in a volatile format) within the live state of the models.
- This allows the models to use this learned information in the next epoch.

On_train_end:

- This subroutine extracts the final model state from the live models, and passes it to the save_data function, as well as reporting said data to the hypertuning apparatus.

```
def run_trial(self, trial, *args, **kwargs):
    """
    Calls methods to build model, tuner, callbacks and begin training

    This method is copied from the superclass methods run_trial and _build_and_fit_model
    the two methods have been combined into one with some additional code added to test if
    the program has been reloaded after being interrupted previously. If it has then the model
    and optimiser states are retrieved and the loaded back into the current trial. Training
    then resumes at the last fully completed epoch

    :param trial: The current trial
    :param args: See superclass method for more information
    :param kwargs: See superclass method for more information
    :return: The return value of `model.fit(), a dictionary or a float
    """

    hp = trial.hyperparameters
    model = self._try_build(hp)
    save_directory = os.path.join(self.get_trial_dir(trial.trial_id), "saves")
    if self.reloaded:
        if os.path.exists(save_directory):
            epoch = self.find_latest_epoch(save_directory)

            with open(os.path.join(save_directory, f"generator_optimiser_epoch_{epoch}.pkl"), 'rb') as f:
                generator_optimizer_config = pickle.load(f)
            model.generator_optimizer = Adam(**generator_optimizer_config)

            with open(os.path.join(save_directory, f"discriminator_optimiser_epoch_{epoch}.pkl"), 'rb') as f:
                discriminator_optimizer_config = pickle.load(f)
            model.discriminator_optimizer = Adam(**discriminator_optimizer_config)

            model.generator.load_weights(os.path.join(save_directory, f"generator_epoch_{epoch}.h5"))
            model.discriminator.load_weights(os.path.join(save_directory, f"discriminator_epoch_{epoch}.h5"))

            kwargs['epochs'] -= epoch
            print("Loaded model weights")
            print(f"Resuming at the end of epoch: {epoch}")
            print(f"Remaining epochs: {kwargs['epochs']}")
            self.reloaded = False
    self.save()
    model_checkpoint = MyCallback(save_directory)
    original_callbacks = kwargs.pop("callbacks", [])
    copied_kwargs = copy.copy(kwargs)
    callbacks = self._deepcopy_callbacks(original_callbacks)
    self._configure_tensorboard_dir(callbacks, trial, 0)
    callbacks.append(tuner_utils.TunerCallback(self, trial))
    callbacks.append(model_checkpoint)
    copied_kwargs["callbacks"] = callbacks
    results = self.hypermodel.fit(hp, model, *args, **copied_kwargs)
    return results
```

This script serves as the models hypertuner by accepting trial sets of hyperparameters as input, then calling all of the above methods together with the in build fit method (which runs the train_step method on all of the data in the database, n many times) in order to facilitate the production of correct output.

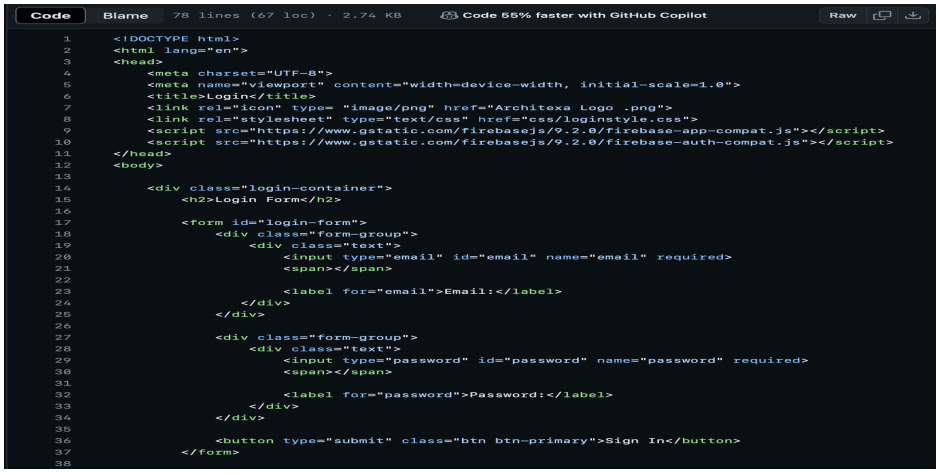
Note that it also contains functionality to retrieve the current weights of the model from a file and reload them, in the event that the model is interrupted mid training.

Front End - Website Development

Login page (index.html)
This is the first page that is presented to the user. The user must enter his/her email and password to access the website, if they are new then they should click on the sign up button to register. Furthermore, if the user forgets his/her password, they should click on the forgot password button, which will redirect them to the forgot password page to make a new one. In the JavaScript section we incorporated Firebase's user authentication feature to verify and store user data.

Features

- Verify user data to log in
- Sign up button redirects to sign up page
- Forgot password button redirects to forgot password page.




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="icon" type= "image/png" href="Architexa Logo .png">
    <link rel="stylesheet" type="text/css" href="css/loginstyle.css">
    <script src="https://www.gstatic.com/firebasejs/9.2.0/firebase-app-compat.js"></script>
    <script src="https://www.gstatic.com/firebasejs/9.2.0/firebase-auth-compat.js"></script>
</head>
<body>

    <div class="login-container">
        <h2>Login Form</h2>

        <form id="login-form">
            <div class="form-group">
                <div class="text">
                    <input type="email" id="email" name="email" required>
                    <span></span>

                    <label for="email">Email:</label>
                </div>
            </div>

            <div class="form-group">
                <div class="text">
                    <input type="password" id="password" name="password" required>
                    <span></span>

                    <label for="password">Password:</label>
                </div>
            </div>

            <button type="submit" class="btn btn-primary">Sign In</button>
        </form>
```

```
            <button type="submit" class="btn btn-primary">Sign In</button>
        </form>

        <div class="additional-links">
            <a href="/login/ForgotPassword.html">Forgot Password</a>
            <a href="/login/Signup.html">Sign Up Here</a>
        </div>
    </div>

    <script>
        const firebaseConfig = {
            apiKey: "AIzaSyDkBZ3wI87PrjdoV7yYhYcXUXJ_7irT4Cs",
            authDomain: "architexa-1f4ce.firebaseapp.com",
            projectId: "architexa-1f4ce",
            storageBucket: "architexa-1f4ce.appspot.com",
            messagingSenderId: "334635524502",
            appId: "1:334635524502:web:a6bfe1993cc6a2eea7fdb7"
        };

        const app = firebase.initializeApp(firebaseConfig);
        const auth = firebase.auth();

        const loginForm = document.getElementById('login-form');
        loginForm.addEventListener('submit', (e) => {
            e.preventDefault();
            const email = loginForm['email'].value;
            const password = loginForm['password'].value;

            auth.signInWithEmailAndPassword(email, password)
            .then((userCredential) => {
                alert('Sign in successful!');
                window.location.href = 'pages/home.html'; // Redirect to home page
            })
            .catch((error) => {
                const errorCode = error.code;
```

```
            const app = firebase.initializeApp(firebaseConfig);
            const auth = firebase.auth();

            const loginForm = document.getElementById('login-form');
            loginForm.addEventListener('submit', (e) => {
                e.preventDefault();
                const email = loginForm['email'].value;
                const password = loginForm['password'].value;

                auth.signInWithEmailAndPassword(email, password)
                .then((userCredential) => {
                    alert('Sign in successful');
                    window.location.href = 'pages/home.html'; // Redirect to home page
                })
                .catch((error) => {
                    const errorCode = error.code;
                    const errorMessage = error.message;
                    // Display a pop-up message for incorrect email/password
                    alert('Incorrect email/password. Please try again.');
                });
            });
        </script>
    </body>
    </html>
```

Home page (home.html)

The html file below is the main page that will serve the user. It is also where the user can write his/her prompt and submit it to the AI model and see their generated image displayed. Furthermore, it also serves as the starting point where the user could navigate to the other pages from. The JavaScript file below linked to this home.html is the API endpoint that sends the POST request to the AI model. The inline script at the bottom is used for dynamic interaction, specifically the sidebar movement. When the sidebar menu button(btn) is clicked the side bar will open(widen) on the screen.

Input

- Prompt. For example, red house.

Output

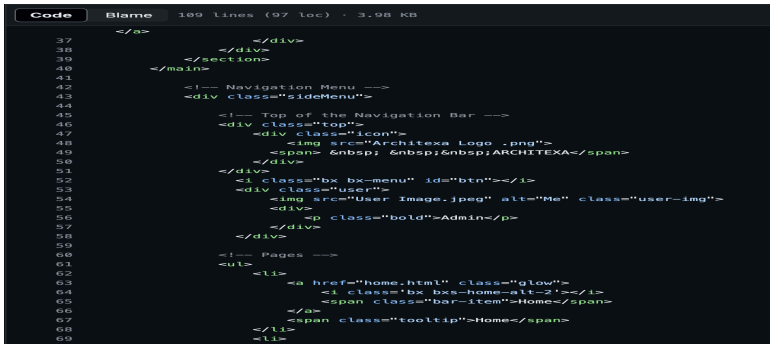
- AI generated image of red house

Features

- Sidebar navigation to about, terms of service, and logout.pages.
- Where AI image will be displayed
- Animated side menu

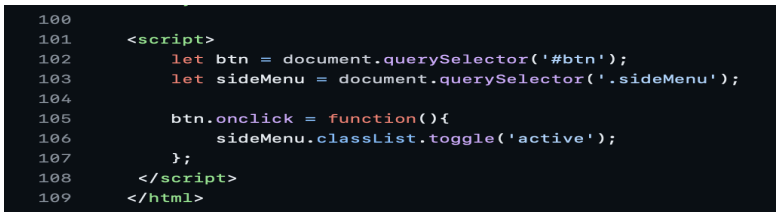
Code Blame 109 lines (97 loc) · 3.98 KB Code 55% faster with GitHub Copilot Raw

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Architexa - Architectural Design Visualization</title>
    <link rel="icon" type= "image/png" href="Architexa Logo .png">
    <link rel="stylesheet" href="../css/styles.css">
    <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>

</head>
<body>
 <div id="wrapper">

    <!-- Input prompt section ( Reseaon being up here is the API call seems can only work like this)-->
    <form action="">
        <input type="text" id="prompt" placeholder="Enter prompt">
        <button id="generate-button" class="submitButton ">
         <i class='bx bx-up-arrow-alt bx-md'></i>
        </button>
    </form>

    <main id="content">
        <header>
            <h1>Welcome to Architexa</h1>
        </header>

        <section id="design-prompt-section">
            <h2>Create Your Architectural Design</h2>
            <!-- Image container for API output -->
            <div id="imageContainer" class="imageDispay">

                <div class=" imageAPI" >
                 <!-- Image first dislayed to user for aesthetic  -->
                  <img src="https://images.unsplash.com/photo-1611602132416-da2045990f76?q=80&w=3087&auto=format&fit=crop&ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D" alt=" Blue house">
                    <a href=" https://unsplash.com/photos/white-and-brown-wooden-house-near-green-trees-during-daytime-u-Hb93V6IWI" target=" _blank"> Blue
House </a>
```



```
Code  Blame  109 lines (97 loc) · 3.98 KB
      </a>
                        </div>
                    </div>
                </section>
            </main>

                <!-- Navigation Menu -->
                <div class="sideMenu">

                    <!-- Top of the Navigation Bar -->
                    <div class="top">
                        <div class="icon">
                            <img src="Architexa Logo .png">
                            <span>     ARCHITEXA</span>
                        </div>
                    </div>
                    <i class="bx bx-menu" id="btn"></i>
                    <div class="user">
                        <img src="User Image.jpeg" alt="Me" class="user-img">
                        <div>
                            <p class="bold">Admin</p>
                        </div>
                    </div>

                    <!-- Pages -->
                    <ul>
                        <li>
                            <a href="home.html" class="glow">
                                <i class='bx bxs-home-alt-2'></i>
                                <span class="bar-item">Home</span>
                            </a>
                            <span class="tooltip">Home</span>
                        </li>
                        <li>
```

```
                <li>
                    <a href="/pages/about.html">
                        <i class='bx bxs-detail'></i>
                        <span class="bar-item">About Us</span>
                    </a>
                    <span class="tooltip">About Us</span>
                </li>
                <li>
                    <a href="/pages/terms.html">
                        <i class='bx bxs-book-open'></i>
                        <span class="bar-item">Terms of service</span>
                    </a>
                    <span class="tooltip">Terms of service</span>
                </li>
                <li>
                    <a href="../index.html">
                        <i class='bx bx-log-out'></i>
                        <span class="bar-item">Log out</span>
                    </a>
                    <span class="tooltip">Log out</span>
                </li>
            </ul>
        </div>
    </div>
        <!-- Main Content Area -->

    <!-- Ensure the path to your main.js is correct and include type="module" for ES6 module support -->
    <script src="../js/script.js"></script>
    <!-- <script src="main.js" type="module"></script> -->

</body>

<script>
    let btn = document.querySelector('#btn');
    let sideMenu = document.querySelector('.sideMenu');
```



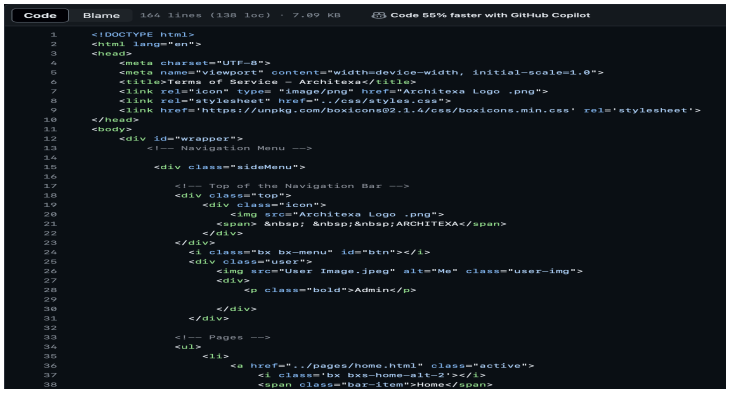
```
<script>
    let btn = document.querySelector('#btn');
    let sideMenu = document.querySelector('.sideMenu');

    btn.onclick = function(){
        sideMenu.classList.toggle('active');
    };
</script>
</html>
```

Terms of Service page (terms.html)

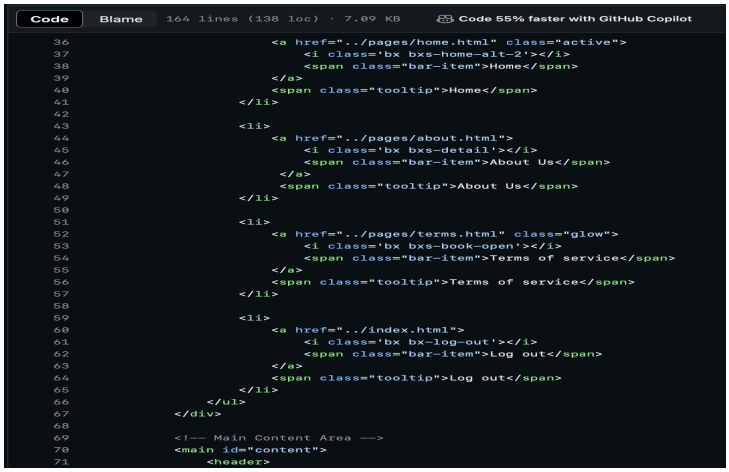
The html file below is the terms of service page, which contains information about the guidelines and rules that users must follow to use our services. The inline script at the bottom is used for dynamic interaction, specifically the sidebar movement. When the sidebar menu button(btn) is clicked the side bar will open(widen) on the screen.

Features

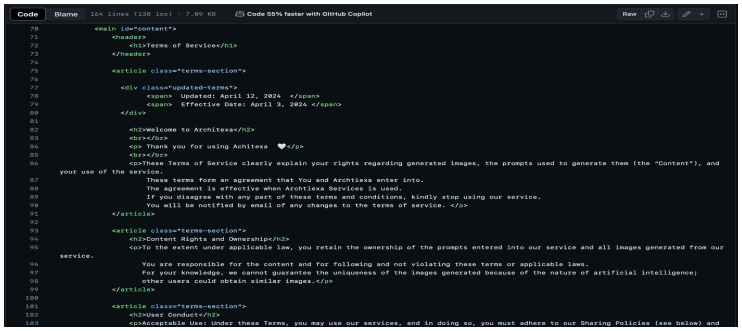
- Sidebar navigation to home, about, and login pages.
- Animated side menu.
- Information about policy and guidelines such as ownership, sharing policy and privacy.



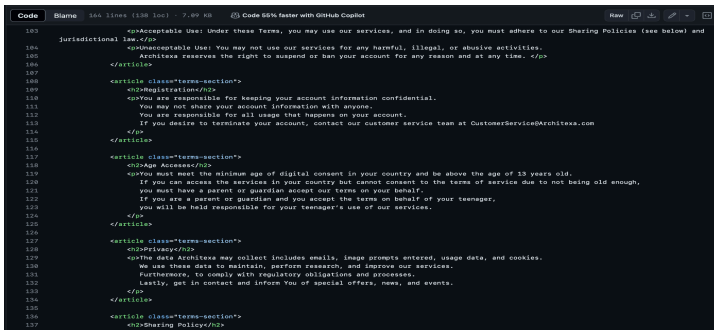
```
Code  Blame  164 lines (138 loc) · 7.09 KB  Code 55% faster with GitHub Copilot
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Terms of Service - Architexa</title>
    <link rel="icon" type= "image/png" href="Architexa Logo .png">
    <link rel="stylesheet" href="../css/styles.css">
    <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>
</head>
<body>
    <div id="wrapper">
        <!-- Navigation Menu -->

        <div class="sideMenu">

            <!-- Top of the Navigation Bar -->
            <div class="top">
                <div class="icon">
                    <img src="Architexa Logo .png">
                    <span>     ARCHITEXA</span>
                </div>
            </div>
            <i class="bx bx-menu" id="btn"></i>
            <div class="user">
                <img src="User Image.jpeg" alt="Me" class="user-img">
                <div>
                    <p class="bold">Admin</p>

                </div>
            </div>

            <!-- Pages -->
            <ul>
                <li>
                    <a href="../pages/home.html" class="active">
                        <i class='bx bxs-home-alt-2'></i>
                        <span class="bar-item">Home</span>
```



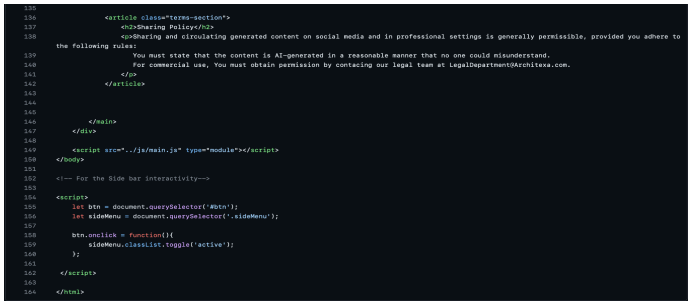

```
                            <a href="../pages/home.html" class="active">
                                <i class='bx bxs-home-alt-2'></i>
                                <span class="bar-item">Home</span>
                            </a>
                            <span class="tooltip">Home</span>
                        </li>

                        <li>
                            <a href="../pages/about.html">
                                <i class='bx bxs-detail'></i>
                                <span class="bar-item">About Us</span>
                             </a>
                             <span class="tooltip">About Us</span>
                        </li>

                        <li>
                            <a href="../pages/terms.html" class="glow">
                                <i class='bx bxs-book-open'></i>
                                <span class="bar-item">Terms of service</span>
                            </a>
                            <span class="tooltip">Terms of service</span>
                        </li>

                        <li>
                            <a href="../index.html">
                                <i class='bx bx-log-out'></i>
                                <span class="bar-item">Log out</span>
                            </a>
                            <span class="tooltip">Log out</span>
                        </li>
                    </ul>
                </div>

                <!-- Main Content Area -->
                <main id="content">
                    <header>
```




```
        <main id="content">
            <header>
                <h1>Terms of Service</h1>
            </header>

            <article class="terms-section">

              <div class="updated-terms">
                    <span>  Updated: April 12, 2024  </span>
                    <span>  Effective Date: April 3, 2024 </span>
              </div>

                <h2>Welcome to Architexa</h2>
                <br></br>
                <p> Thank you for using Achitexa  ❤</p>
                <br></br>
                <p>These Terms of Service clearly explain your rights regarding generated images, the prompts used to generate them (the "Content"), and your use of the service.
                    These terms form an agreement that You and Architexa enter into.
                    The agreement is effective when Architexa Services is used.
                    If you disagree with any part of these terms and conditions, kindly stop using our service.
                    You will be notified by email of any changes to the terms of service. </p>
            </article>

            <article class="terms-section">
                <h2>Content Rights and Ownership</h2>
                <p>To the extent under applicable law, you retain the ownership of the prompts entered into our service and all images generated from our service.
                    You are responsible for the content and for following and not violating these terms or applicable laws.
                    For your knowledge, we cannot guarantee the uniqueness of the images generated because of the nature of artificial intelligence;
                    other users could obtain similar images.</p>
            </article>

            <article class="terms-section">
                <h2>User Conduct</h2>
                <p>Acceptable Use: Under these Terms, you may use our services, and in doing so, you must adhere to our Sharing Policies (see below) and
```




```
                <p>Acceptable Use: Under these Terms, you may use our services, and in doing so, you must adhere to our Sharing Policies (see below) and jurisdictional law.</p>
                <p>Unacceptable Use: You may not use our services for any harmful, illegal, or abusive activities.
                    Architexa reserves the right to suspend or ban your account for any reason and at any time. </p>
            </article>

            <article class="terms-section">
                <h2>Registration</h2>
                <p>You are responsible for keeping your account information confidential.
                    You may not share your account information with anyone.
                    You are responsible for all usage that happens on your account.
                    If you desire to terminate your account, contact our customer service team at CustomerService@Architexa.com
                </p>
            </article>

            <article class="terms-section">
                <h2>Age Accesses</h2>
                <p>You must meet the minimum age of digital consent in your country and be above the age of 13 years old.
                    If you can access the services in your country but cannot consent to the terms of service due to not being old enough,
                    you must have a parent or guardian accept our terms on your behalf.
                    If you are a parent or guardian and you accept the terms on behalf of your teenager,
                    you will be held responsible for your teenager's use of our services.
                </p>
            </article>

            <article class="terms-section">
                <h2>Privacy</h2>
                <p>The data Architexa may collect includes emails, image prompts entered, usage data, and cookies.
                    We use these data to maintain, perform research, and improve our services.
                    Furthermore, to comply with regulatory obligations and processes.
                    Lastly, get in contact and inform You of special offers, news, and events.
                </p>
            </article>

            <article class="terms-section">
                <h2>Sharing Policy</h2>
```



```
            <article class="terms-section">
                <h2>Sharing Policy</h2>
                <p>Sharing and circulating generated content on social media and in professional settings is generally permissible, provided you adhere to the following rules:
                    You must state that the content is AI-generated in a reasonable manner that no one could misunderstand.
                    For commercial use, You must obtain permission by contacing our legal team at LegalDepartment@Architexa.com.
                </p>
            </article>



        </main>
    </div>

    <script src="../js/main.js" type="module"></script>
</body>

<!-- For the Side bar interactivity-->

<script>
    let btn = document.querySelector('#btn');
    let sideMenu = document.querySelector('.sideMenu');

    btn.onclick = function(){
        sideMenu.classList.toggle('active');
    };

</script>

</html>
```

About page (about.html)

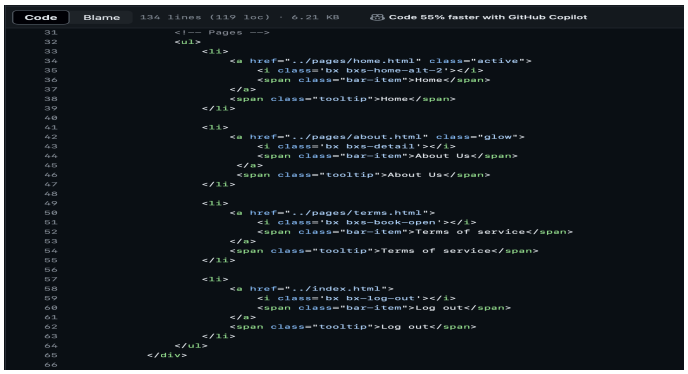
The html file below is the about page, where information about the project, team, and our aims is written. The inline script at the bottom is used for dynamic interaction, specifically the sidebar movement. When the sidebar menu button(btn) is clicked the side bar will open(widen) on the screen.

Features

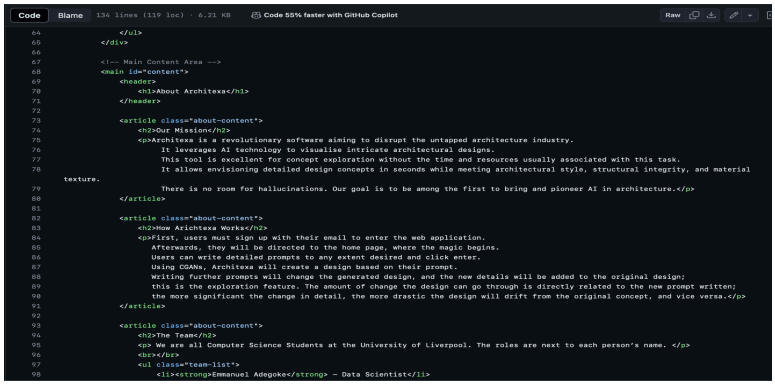
- Sidebar navigation to home, terms of service, and login pages.
- Animated side menu

- Information about how Architexa works, the team behind the project, and future plans.

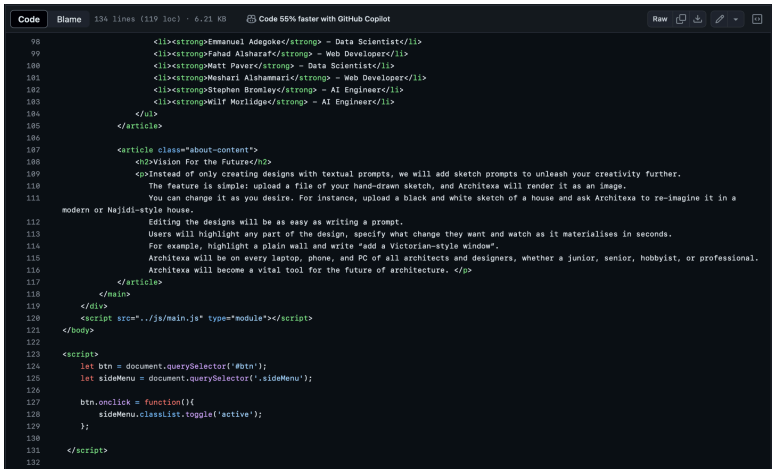
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About Architexa</title>
    <link rel="icon" type= "image/png" href="Architexa Logo .png">
    <link rel="stylesheet" href="../css/styles.css">
    <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>
</head>
<body>
    <div id="wrapper">
        <!-- Navigation Menu -->
        <div class="sideMenu">

            <!-- Top of the Navigation Bar -->
            <div class="top">
                <div class="icon">
                    <img src="Architexa Logo .png">
                  <span>     ARCHITEXA</span>
                </div>
            </div>
              <i class="bx bx-menu" id="btn"></i>
              <div class="user">
                  <img src="User Image.jpeg" alt="Me" class="user-img">
                  <div>
                      <p class="bold">Admin</p>
                  </div>
              </div>

            <!-- Pages -->
            <ul>
```



```
Code  Blame  134 lines (119 loc) · 6.21 KB  Code 55% faster with GitHub Copilot
            <!-- Pages -->
            <ul>
                <li>
                    <a href="../pages/home.html" class="active">
                        <i class='bx bxs-home-alt-2'></i>
                        <span class="bar-item">Home</span>
                    </a>
                    <span class="tooltip">Home</span>
                </li>

                <li>
                    <a href="../pages/about.html" class="glow">
                        <i class='bx bxs-detail'></i>
                        <span class="bar-item">About Us</span>
                    </a>
                    <span class="tooltip">About Us</span>
                </li>

                <li>
                    <a href="../pages/terms.html">
                        <i class='bx bxs-book-open'></i>
                        <span class="bar-item">Terms of service</span>
                    </a>
                    <span class="tooltip">Terms of service</span>
                </li>

                <li>
                    <a href="../index.html">
                        <i class='bx bx-log-out'></i>
                        <span class="bar-item">Log out</span>
                    </a>
                    <span class="tooltip">Log out</span>
                </li>
            </ul>
        </div>
```



```
Code  Blame  134 lines (119 loc) · 6.21 KB  Code 55% faster with GitHub Copilot  Raw
            </ul>
        </div>

        <!-- Main Content Area -->
        <main id="content">
            <header>
                <h1>About Architexa</h1>
            </header>

            <article class="about-content">
                <h2>Our Missions</h2>
                <p>Architexa is a revolutionary software aiming to disrupt the untapped architecture industry.
                    It leverages AI technology to visualise intricate architectural designs.
                    This tool is excellent for concept exploration without the time and resources usually associated with this task.
                    It allows envisioning detailed design concepts in seconds while meeting architectural style, structural integrity, and material
texture.
                    There is no room for hallucinations. Our goal is to be among the first to bring and pioneer AI in architecture.</p>
            </article>

            <article class="about-content">
                <h2>How Arichtexa Works</h2>
                <p>First, users must sign up with their email to enter the web application.
                   Afterwards, they will be directed to the home page, where the magic begins.
                   Users can write detailed prompts to any extent desired and click enter.
                   Using CGANs, Architexa will create a design based on their prompt.
                   Writing further prompts will change the generated design, and the new details will be added to the original design;
                   this is the exploration feature. The amount of change the design can go through is directly related to the new prompt written;
                   the more significant the change in detail, the more drastic the design will drift from the original concept, and vice versa.</p>
            </article>

            <article class="about-content">
                <h2>The Team</h2>
                <p> We are all Computer Science Students at the University of Liverpool. The roles are next to each person's name. </p>
                <br></br>
                <ul class="team-list">
                    <li><strong>Emmanuel Adegoke</strong> - Data Scientist</li>
```



```
Code  Blame  134 lines (119 loc) · 6.21 KB  Code 55% faster with GitHub Copilot  Raw
                    <li><strong>Emmanuel Adegoke</strong> - Data Scientist</li>
                    <li><strong>Fahad Alsharaf</strong> - Web Developer</li>
                    <li><strong>Matt Paver</strong> - Data Scientist</li>
                    <li><strong>Meshari Alshammari</strong> - Web Developer</li>
                    <li><strong>Stephen Bromley</strong> - AI Engineer</li>
                    <li><strong>Wilf Morlidge</strong> - AI Engineer</li>
                </ul>
            </article>

            <article class="about-content">
                <h2>Vision For the Future</h2>
                <p>Instead of only creating designs with textual prompts, we will add sketch prompts to unleash your creativity further.
                   The feature is simple: upload a file of your hand-drawn sketch, and Architexa will render it as an image.
                   You can change it as you desire. For instance, upload a black and white sketch of a house and ask Architexa to re-imagine it in a
modern or Najidi-style house.
                   Editing the designs will be as easy as writing a prompt.
                   Users will highlight any part of the design, specify what change they want and watch as it materialises in seconds.
                   For example, highlight a plain wall and write "add a Victorian-style window".
                   Architexa will be on every laptop, phone, and PC of all architects and designers, whether a junior, senior, hobbyist, or professional.
                   Architexa will become a vital tool for the future of architecture. </p>
            </article>
        </main>
    </div>
    <script src="../js/main.js" type="module"></script>
</body>

<script>
    let btn = document.querySelector('#btn');
    let sideMenu = document.querySelector('.sideMenu');

    btn.onclick = function(){
        sideMenu.classList.toggle('active');
    };

</script>
```

Sign up page (Signup.html)

The file below is the sign up page. Its function is to register new users by taking their email and passwords as input and storing them in a Firebase database. After signing up the user information is saved on the Firebase authentication database. Thus, every time the same user wants to log in he only needs to provide his/her email and password.

Input

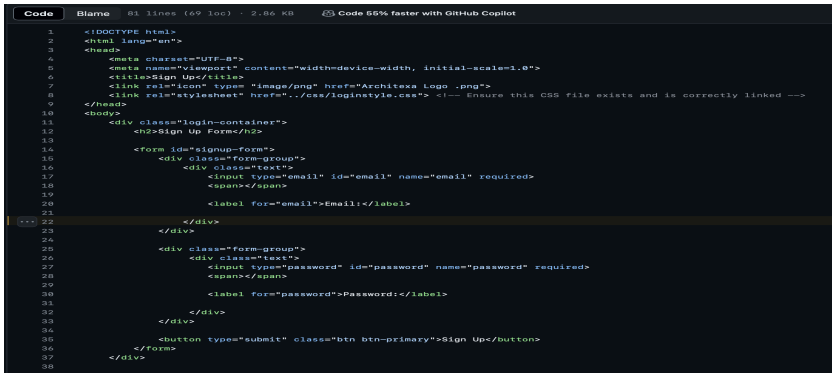
- Email and password

Firebase Processing and Storing User Data

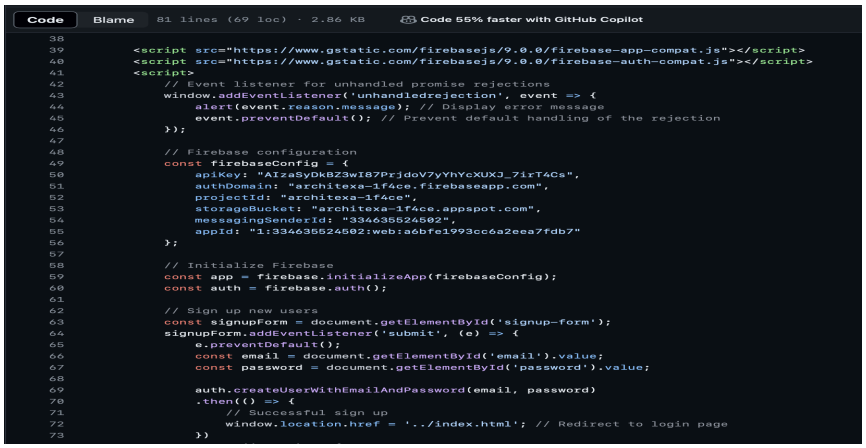
- User email and password

Output

- Redirects the user to the login page to re-enter email and password.



```
Code  Blame  81 lines (69 loc) · 2.86 KB  Code 55% faster with GitHub Copilot

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sign Up</title>
    <link rel="icon" type= "image/png" href="Architexa Logo .png">
    <link rel="stylesheet" href="../css/loginstyle.css"> <!-- Ensure this CSS file exists and is correctly linked -->
</head>
<body>
    <div class="login-container">
        <h2>Sign Up Form</h2>

        <form id="signup-form">
            <div class="form-group">
                <div class="text">
                    <input type="email" id="email" name="email" required>
                    <span></span>

                    <label for="email">Email:</label>

                </div>
            </div>

            <div class="form-group">
                <div class="text">
                    <input type="password" id="password" name="password" required>
                    <span></span>

                    <label for="password">Password:</label>

                </div>
            </div>

            <button type="submit" class="btn btn-primary">Sign Up</button>
        </form>
    </div>
```



```
Code  Blame  81 lines (69 loc) · 2.86 KB  Code 55% faster with GitHub Copilot

    <script src="https://www.gstatic.com/firebasejs/9.0.0/firebase-app-compat.js"></script>
    <script src="https://www.gstatic.com/firebasejs/9.0.0/firebase-auth-compat.js"></script>
    <script>
        // Event listener for unhandled promise rejections
        window.addEventListener('unhandledrejection', event => {
            alert(event.reason.message); // Display error message
            event.preventDefault(); // Prevent default handling of the rejection
        });

        // Firebase configuration
        const firebaseConfig = {
            apiKey: "AIzaSyDkB23wI87PrjdoV7yYhYcXUXJ_7irT4Cs",
            authDomain: "architexa-1f4ce.firebaseapp.com",
            projectId: "architexa-1f4ce",
            storageBucket: "architexa-1f4ce.appspot.com",
            messagingSenderId: "334635524502",
            appId: "1:334635524502:web:a6bfe1993cc6a2eea7fdb7"
        };

        // Initialize Firebase
        const app = firebase.initializeApp(firebaseConfig);
        const auth = firebase.auth();

        // Sign up new users
        const signupForm = document.getElementById('signup-form');
        signupForm.addEventListener('submit', (e) => {
            e.preventDefault();
            const email = document.getElementById('email').value;
            const password = document.getElementById('password').value;

            auth.createUserWithEmailAndPassword(email, password)
            .then(() => {
                // Successful sign up
                window.location.href = '../index.html'; // Redirect to login page
            })
```

```
                auth.createUserWithEmailAndPassword(email, password)
                .then(() => {
                    // Successful sign up
                    window.location.href = '../index.html'; // Redirect to login page
                })
                .catch((error) => {
                    // Handle errors
                    alert(error.message); // Display error message
                });
            });
        </script>
    </body>
    </html>
```

Forgot Password page (ForgotPassword.html)

The file below is the forgot password page. Its function is to reset the user's password if they have forgotten/ lost it. That is done by taking the user's registered email and sending them a link that will direct them to a page with an input box to write their new password in.

Input

- Email

Firebase Send link via email

- Input box to write new password

Output

- New password saved and can be used to login

Code Blame 66 lines (58 loc) · 2.39 KB Code 55% faster with GitHub Copilot

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Forgot Password</title>
    <link rel="icon" type "image/png" href="Architexa Logo .png">
    <link rel="stylesheet" href="../css/loginstyle.css"> <!-- Link to your CSS file for styling -->
</head>
<body>
    <div class="login-container">
        <h2>Reset Password</h2>

        <form id="reset-form">
            <div class="form-group">
                <div class="text">
                    <input type="email" id="email" name="email" required>
                    <span></span>

                    <label for="email">Email:</label>

                </div>
            </div>

            <button type="submit" class="btn btn-primary">Reset Password</button>
        </form>
    </div>

    <script src="https://www.gstatic.com/firebasejs/9.0.0/firebase-app-compat.js"></script>
    <script src="https://www.gstatic.com/firebasejs/9.0.0/firebase-auth-compat.js"></script>
    <script>
        // Your Firebase configuration
        const firebaseConfig = {
            apiKey: "AIzaSyDkBZ3wI87PrjdoV7yYhYcXUXJ_7irT4Cs",
            authDomain: "architexa-1f4ce.firebaseapp.com",
            projectId: "architexa-1f4ce",
            storageBucket: "architexa-1f4ce.appspot.com",
            messagingSenderId: "334635524502",
```

Code Blame 66 lines (58 loc) · 2.39 KB Code 55% faster with GitHub Copilot

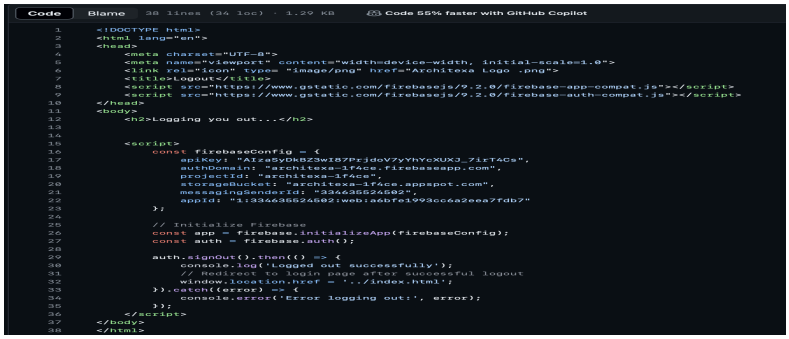
```
        const firebaseConfig = {
            apiKey: "AIzaSyDkBZ3wI87PrjdoV7yYhYcXUXJ_7irT4Cs",
            authDomain: "architexa-1f4ce.firebaseapp.com",
            projectId: "architexa-1f4ce",
            storageBucket: "architexa-1f4ce.appspot.com",
            messagingSenderId: "334635524502",
            appId: "1:334635524502:web:a6bfe1993cc6a2eea7fdb7"
        };

        // Initialize Firebase
        const app = firebase.initializeApp(firebaseConfig);
        const auth = firebase.auth();

        // Reset password
        const resetForm = document.getElementById('reset-form');
        resetForm.addEventListener('submit', (e) => {
            e.preventDefault();
            const email = resetForm['email'].value;

            auth.sendPasswordResetEmail(email)
            .then(() => {
                console.log('Password reset email sent.');
                alert('Password reset email sent. Please check your inbox.'); // Notify the user
                window.location.href = '../index.html'; // Redirect to login page
            })
            .catch((error) => {
                const errorCode = error.code;
                const errorMessage = error.message;
                alert(errorMessage); // Display an error message to the user
            });
        });
    </script>
</body>
</html>
```

Log out Screen (Logout .html)

The file below is the logout screen. Its function is to ensure that user sessions are properly terminated. Furthermore, to print “ Logging you out “ on the user screen to notify them that they have successfully been logged out. This is in accordance with user experience etiquette which is to develop a relationship of trust that the users are in control of their login state.

Features

- Ensure user session is terminated
- Secure logout process to protect user data
- Display clear and concise logout message



```
Code  Blame  38 lines (34 loc) · 1.29 KB  Code 55% faster with GitHub Copilot

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="icon" type= "image/png" href="Architexa Logo .png">
    <title>Logout</title>
    <script src="https://www.gstatic.com/firebasejs/9.2.0/firebase-app-compat.js"></script>
    <script src="https://www.gstatic.com/firebasejs/9.2.0/firebase-auth-compat.js"></script>
</head>
<body>
    <h2>Logging you out...</h2>

    <script>
        const firebaseConfig = {
            apiKey: "AIzaSyDkBZ3wI87PrjdoV7yYhYcXUXJ_7irT4Cs",
            authDomain: "architexa-1f4cs.firebaseapp.com",
            projectId: "architexa-1f4cs",
            storageBucket: "architexa-1f4cs.appspot.com",
            messagingSenderId: "324638524682",
            appId: "1:324638524682:web:aabfe1993cc6a2eea7fdb7"
        };

        // Initialize Firebase
        const app = firebase.initializeApp(firebaseConfig);
        const auth = firebase.auth();

        auth.signOut().then(() => {
            console.log('Logged out successfully');
            // Redirect to login page after successful logout
            window.location.href = '../index.html';
        }).catch((error) => {
            console.error('Error logging out:', error);
        });
    </script>
</body>
</html>
```

API Endpoint (script .js)

The file below is the website API endpoint. Its function is to send the user prompt as a POST request to the flask API endpoint to generate an image. Retrieve the generated image and then display it on the user’s screen in a tidy manner. Furthermore, it validates if the user entered a prompt or not.

Input

- User prompt

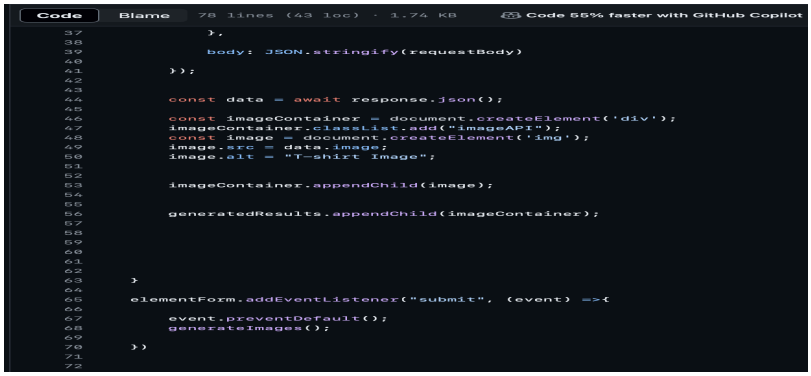
Send to Flask API

- generateImage() is called to generate an image

Output

- Generated image is sent back and displayed to the user

```
Code  Blame  78 lines (43 loc) · 1.74 KB  Code 55% faster with GitHub Copilot

const elementForm = document.querySelector("form");
const elementPrompt = document.getElementById("prompt");
const generatedResults = document.querySelector(".imageDispay");

let dataInput =  " ";

async function generateImages(){

    generatedResults.innerHTML = '';

    // Make sure input is not empty
    dataInput = elementPrompt.value.trim();
    if (dataInput === ''){
        wni
        console.log(" Empty input")

        return;
    }

    dataInput = elementPrompt.value;
    const dynamicURL = `https://that-flaskap1.onrender.com/hi`;
    const requestBody = {
        prompt: dataInput

    };


    console.log(dynamicURL)

    const response = await fetch(dynamicURL, {
        method: "POST",
        headers: {
            'Content-Type' : 'application/json'
        },
```



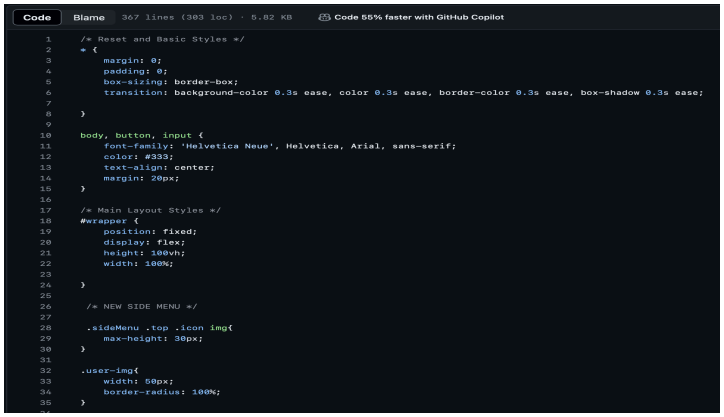

```
            },
            body: JSON.stringify(requestBody)
        });

        const data = await response.json();

        const imageContainer = document.createElement('div');
        imageContainer.classList.add("imageAPI");
        const image = document.createElement('img');
        image.src = data.image;
        image.alt = "T-shirt Image";


        imageContainer.appendChild(image);


        generatedResults.appendChild(imageContainer);




    }

    elementForm.addEventListener("submit", (event) =>{
        event.preventDefault();
        generateImages();
    })
```

UI Design (styles.css)

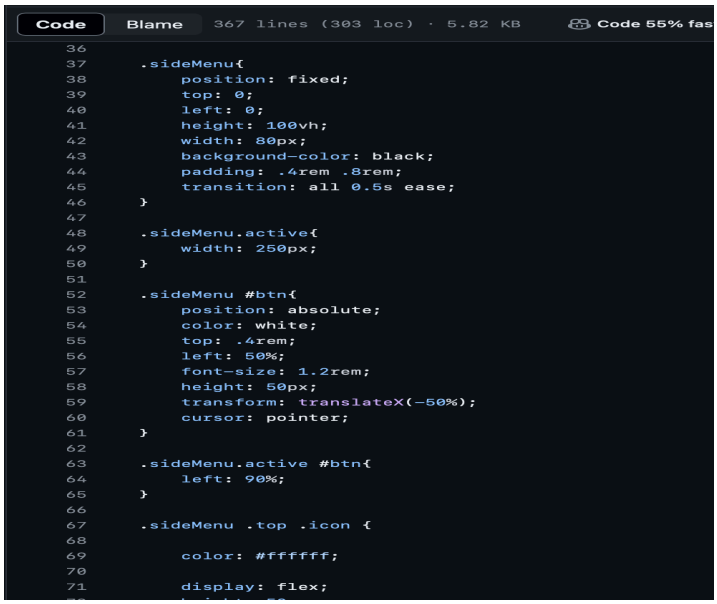
The file below is the CSS file used to give Architexa a sleek minimalistic look. It is separated into five sections, main layout, side menu, content area, generated image display, and shared sections for the about.html and terms.html design.

Features

- Minimalistic/ futuristic style
- Professional branding look




```
/* Reset and Basic Styles */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    transition: background-color 0.3s ease, color 0.3s ease, border-color 0.3s ease, box-shadow 0.3s ease;

}

body, button, input {
    font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif;
    color: #333;
    text-align: center;
    margin: 20px;
}

/* Main Layout Styles */
#wrapper {
    position: fixed;
    display: flex;
    height: 100vh;
    width: 100%;

}

 /* NEW SIDE MENU */

.sideMenu .top .icon img{
    max-height: 30px;
}
.user-img{
    width: 50px;
    border-radius: 100%;
}
```




```
    .sideMenu{
        position: fixed;
        top: 0;
        left: 0;
        height: 100vh;
        width: 80px;
        background-color: black;
        padding: .4rem .8rem;
        transition: all 0.5s ease;
    }

    .sideMenu.active{
        width: 250px;
    }

    .sideMenu #btn{
        position: absolute;
        color: white;
        top: .4rem;
        left: 50%;
        font-size: 1.2rem;
        height: 50px;
        transform: translateX(-50%);
        cursor: pointer;
    }

    .sideMenu.active #btn{
        left: 90%;
    }

    .sideMenu .top .icon {

        color: #ffffff;

        display: flex;
```

Code Blame 367 lines (303 loc) · 5.82 KB Code

```
70
71          display: flex;
72          height: 50px;
73          width: 100px;
74          align-items: center;
75          pointer-events: none;
76          opacity: 0;
77      }
78
79      .sideMenu.active .top .icon{
80          opacity: 1;
81      }
82
83      .top .icon i{
84          font-size: 2rem;
85          margin-right: 5px;
86      }
87
88      .user{
89          display: flex;
90          align-items: center;
91          margin: 1rem 0;
92      }
93
94      .user p {
95          color: white;
96          opacity: 1;
97          margin-left: 1rem;
98      }
99
100     .sideMenu p{
101         opacity: 0;
102     }
103
104     .sideMenu.active p{
105
```

Code Blame 367 lines (303 loc) · 5.

```
104     .sideMenu.active p{
105
106         opacity: 1;
107     }
108
109     .sideMenu ul li{
110         position: relative;
111         list-style-type: none;
112         height: 50%;
113         width: 90%;
114         margin: 0.8rem auto;
115         line-height: 50px;
116     }
117
118     .sideMenu ul li a{
119         color: white;
120         display: flex;
121         align-items: center;
122         text-decoration: none;
123         border-radius: 0.8rem;
124     }
125
126     .sideMenu ul li a:hover {
127         background-color: white;
128         color: black;
129     }
130
131     .sideMenu ul li .glow {
132         background-color: white;
133         color: black;
134     }
135
136     .sideMenu ul li a i{
137         min-width: 50px;
138         text-align: center;
139         height: 50px;
```

Code Blame 367 lines (303 loc) · 5.82 KB

```
139         height: 50px;
140         border-radius: 12px;
141         line-height: 50px;
142     }
143
144     .sideMenu .bar-item{
145         opacity: 0;
146     }
147
148     .sideMenu.active .bar-item{
149         opacity: 1;
150     }
151
152     .sideMenu ul li .tooltip{
153         color: white;
154         background-color: black;
155         position: absolute;
156         left: 125px;
157         top: 50%;
158         transform: translate(-50%, -50%);
159         box-shadow: 0 0.05rem 0.08rem black;
160         border-radius: 0.7rem;
161         padding: .4rem 1.2rem;
162         line-height: 1.8rem;
163         z-index: 20;
164         opacity: 0;
165     }
166
167     .sideMenu ul li:hover .tooltip{
168
169         opacity: 1;
170     }
171
172     .sideMenu.active ul li .tooltip{
173         display: none;
174     }
```

Code Blame 367 lines (303 loc) · 5.82 KB

```
173             display: none;
174         }
175
176         /* Content Area */
177         #content {
178             flex-grow: 1;
179             background-color: #f4f4f4;
180             overflow-y: scroll;
181             padding: 25px;
182         }
183
184         header {
185             text-align: center;
186             padding: 30px;
187             background-color: #4A4A4A;
188             color: #FFF;
189         }
190
191         header h1 {
192             font-size: 3rem;
193             font-weight: 300;
194         }
195
196         /* Image Generated Display area*/
197
198         .imageDispay{
199             display: flex;
200             flex-wrap: wrap;
201             max-width: 1400px;
202             margin: 0 auto;
203         }
204
205
206         .imageAPI{
207             margin-bottom: 60px;
208             width: 60%;
```

```
208             width: 60%;
209             height: 100%;
210             border-radius: 5px;
211             overflow: hidden ;
212             box-shadow: 0 0 10px #212020;
213             left: 50%;
214             transform: translateX(30%);
215         }
216
217         .imageAPI img{
218
219             width: 100%;
220             height: 100%;
221             object-fit: cover;
222             transition: opacity 0.1s;
223          }
224
225         .imageAPI :hover{
226             opacity: 0.9;
227
228         }
229
230
231         /*  Prompt Bar Input Styles*/
232
233         #prompt {
234
235             width: 650px;
236             padding: 5px;
237             margin-right: 10px;
238         }
239
240         #prompt   {
241
242             position: absolute;
243             bottom: 0%;
```

```
243         bottom: 0%;
244         left: 30%;
245         border: 1px solid #fff;
246         color: #fff;
247         background-color: #333;
248         padding: 10px;
249         font-size: 16px;
250         margin-right: 10px;
251         border-radius: 5px;
252         z-index: 10;
253     }
254
255
256     .submitButton {
257
258         position: absolute;
259         bottom: 0.5%;
260         right: 22.5%;
261         background-color: #444;
262         color: white;
263         border: 1px solid #444;
264         font-size: 20px;
265         border-radius: 5px;
266         cursor: pointer;
267         z-index: 10;
268         height: 2rem;
269         width: 3rem;
270      }
271
272     /* Shared Section Styles for About and Terms Content */
273     #design-prompt-section, .about-content, .terms-content, .terms-section {
274         text-align: center;
275         padding: 50px;
276         background-color: #fff;
277         border-radius: 15px;
278         box-shadow: 0 5px 15px rgba(0,0,0,0.1);
```

```
278             box-shadow: 0 5px 15px rgba(0,0,0,0.1);
279             margin: 20px auto;
280             max-width: 800px;
281         }
282
283         .updated-terms {
284             justify-content: space-between;
285             display: flex;
286          }
287
288         /* Login Page Styles */
289         .login-container {
290             max-width: 400px;
291             margin: 0 auto;
292             padding: 40px;
293             background-color: #ffffff;
294             border-radius: 10px;
295             box-shadow: 0 8px 16px rgba(0,0,0,0.1);
296             animation: fadeIn 0.5s ease-in-out;
297         }
298
299         .login-form h2 {
300             margin-bottom: 20px;
301             color: #333;
302         }
303
304         .input-group {
305             position: relative;
306             margin-bottom: 25px;
307         }
308
309         .input-group input {
310             width: 100%;
311             padding: 10px;
312             border: 1px solid #ccc;
313             border-radius: 5px;
```

Code Blame 367 lines (303 loc) · 5.82 KB Code 55% faster with GitHub Copilot

```
309     .input-group input {
310         width: 100%;
311         padding: 10px;
312         border: 1px solid #ccc;
313         border-radius: 5px;
314         font-size: 16px;
315     }
316
317     .input-group label {
318         position: absolute;
319         top: 0;
320         left: 10px;
321         padding: 10px 0;
322         font-size: 16px;
323         color: #999;
324         pointer-events: none;
325         transition: transform 0.2s ease-in-out, font-size 0.2s ease-in-out;
326     }
327
328     .input-group input:focus + label,
329     .input-group input:valid + label {
330         transform: translateY(-24px);
331         font-size: 14px;
332         color: #007bff;
333     }
334
335     .login-btn {
336         width: 100%;
337         padding: 12px 20px;
338         background-color: #007bff;
339         color: #ffffff;
340         border: none;
341         border-radius: 5px;
342         cursor: pointer;
343         font-size: 18px;
344         letter-spacing: 1px;
```

Code Blame 367 lines (303 loc) · 5.82 KB Code 55% faste

```
334
335     .login-btn {
336         width: 100%;
337         padding: 12px 20px;
338         background-color: #007bff;
339         color: #ffffff;
340         border: none;
341         border-radius: 5px;
342         cursor: pointer;
343         font-size: 18px;
344         letter-spacing: 1px;
345         transition: background-color 0.3s ease;
346     }
347
348     .login-btn:hover {
349         background-color: #0056b3;
350     }
351
352     /* Animations */
353     @keyframes fadeIn {
354         from {
355             opacity: 0;
356             transform: translateY(20px);
357         }
358         to {
359             opacity: 1;
360             transform: translateY(0);
361         }
362     }
363
364     /* Responsive Design */
365     @media (max-width: 768px) {
366         /* Adjust styles for smaller screens as needed */
367     }
```

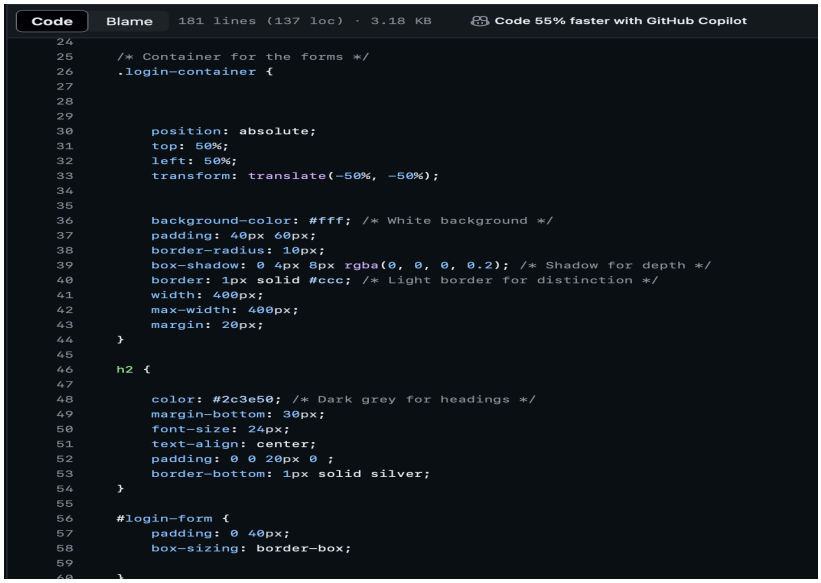
UI Design (loginstyle.css)

The file below is the CSS file used to give the login, sign up, and forgot password pages of Architexa a branded and professional look. The styles were made to make the email and password input animated, to entertain the user's eyes while using the login interface. It also serves as a visual signal to let the user know that his/her input was entered.

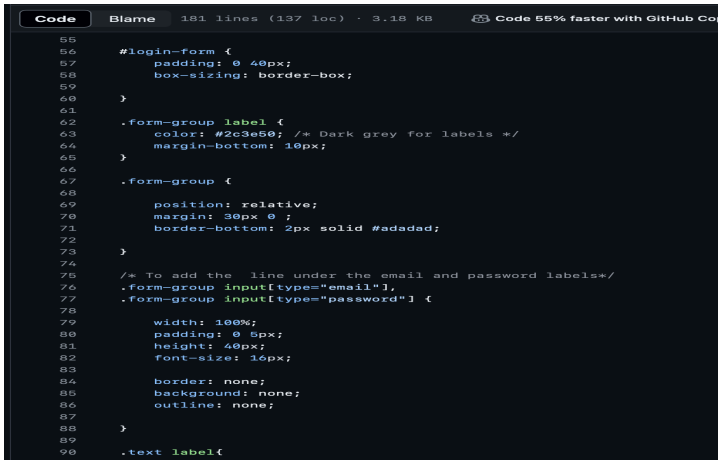
Features

- Animated text input box
- Futuristic style
- Professional branding look

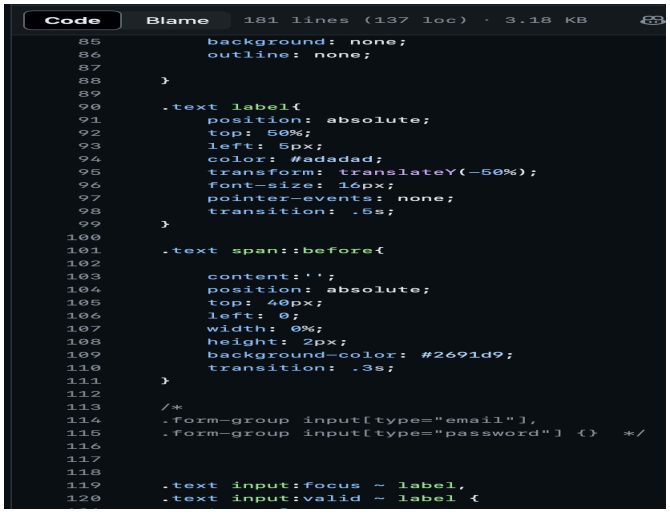

```css
/* General resets and base styles */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

body, html {

    background-image: url('thatDesign.png');
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
    width: 100%;
    height: 100%;
    /* background: linear-gradient(120deg, #2980b9, #8e44ad);  Dark background for contrast */
    display: flex;
    justify-content: center;


}


/* Container for the forms */
.login-container {
```



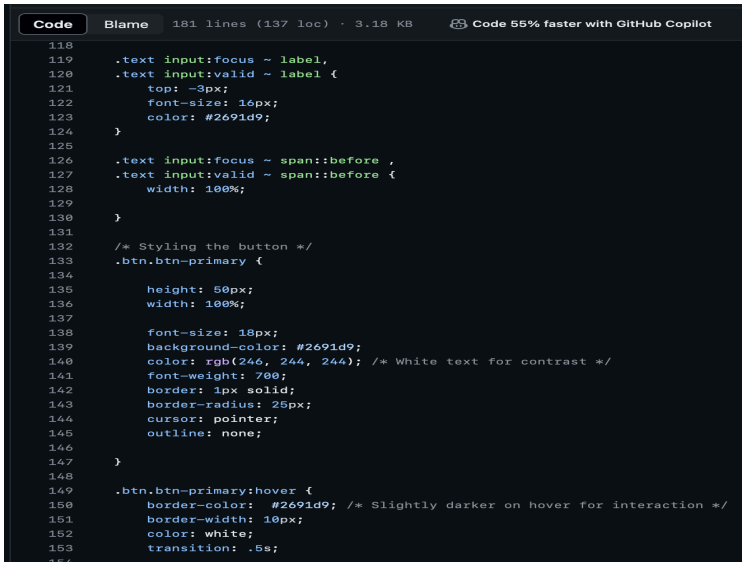

```css
/* Container for the forms */
.login-container {



    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);


    background-color: #fff; /* White background */
    padding: 40px 60px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2); /* Shadow for depth */
    border: 1px solid #ccc; /* Light border for distinction */
    width: 400px;
    max-width: 400px;
    margin: 20px;
}

h2 {

    color: #2c3e50; /* Dark grey for headings */
    margin-bottom: 30px;
    font-size: 24px;
    text-align: center;
    padding: 0 0 20px 0 ;
    border-bottom: 1px solid silver;
}

#login-form {
    padding: 0 40px;
    box-sizing: border-box;

}
```




```
55
56      #login-form {
57          padding: 0 40px;
58          box-sizing: border-box;
59
60      }
61
62      .form-group label {
63          color: #2c3e50; /* Dark grey for labels */
64          margin-bottom: 10px;
65      }
66
67      .form-group {
68
69          position: relative;
70          margin: 30px 0 ;
71          border-bottom: 2px solid #adadad;
72
73      }
74
75      /* To add the  line under the email and password labels*/
76      .form-group input[type="email"],
77      .form-group input[type="password"] {
78
79          width: 100%;
80          padding: 0 5px;
81          height: 40px;
82          font-size: 16px;
83
84          border: none;
85          background: none;
86          outline: none;
87
88      }
89
90      .text label{
```




```
85              background: none;
86              outline: none;
87
88          }
89
90          .text label{
91              position: absolute;
92              top: 50%;
93              left: 5px;
94              color: #adadad;
95              transform: translateY(-50%);
96              font-size: 16px;
97              pointer-events: none;
98              transition: .5s;
99          }
100
101         .text span::before{
102
103             content:'';
104             position: absolute;
105             top: 40px;
106             left: 0;
107             width: 0%;
108             height: 2px;
109             background-color: #2691d9;
110             transition: .3s;
111         }
112
113         /*
114         .form-group input[type="email"],
115         .form-group input[type="password"] {}  */
116
117
118
119         .text input:focus ~ label,
120         .text input:valid ~ label {
```




```
118
119     .text input:focus ~ label,
120     .text input:valid ~ label {
121         top: -3px;
122         font-size: 16px;
123         color: #2691d9;
124     }
125
126     .text input:focus ~ span::before ,
127     .text input:valid ~ span::before {
128         width: 100%;
129
130     }
131
132     /* Styling the button */
133     .btn.btn-primary {
134
135         height: 50px;
136         width: 100%;
137
138         font-size: 18px;
139         background-color: #2691d9;
140         color: rgb(246, 244, 244); /* White text for contrast */
141         font-weight: 700;
142         border: 1px solid;
143         border-radius: 25px;
144         cursor: pointer;
145         outline: none;
146
147     }
148
149     .btn.btn-primary:hover {
150         border-color:  #2691d9; /* Slightly darker on hover for interaction */
151         border-width: 10px;
152         color: white;
153         transition: .5s;
```


```
148
149     .btn.btn-primary:hover {
150         border-color:  #2691d9; /* Slightly darker on hover for interaction */
151         border-width: 10px;
152         color: white;
153         transition: .5s;
154
155     }
156
157     .additional-links {
158
159         justify-content: space-between;
160         display: flex;
161     }
162
163
164     /* Link styles */
165     .additional-links a {
166         color: #2c3e50; /* Dark grey for links */
167         text-decoration: none;
168         margin: 25px 0;
169     }
170
171     .additional-links a:hover {
172         text-decoration: underline;
173     }
174
175     /* Responsive design for smaller screens */
176     @media (max-width: 480px) {
177         .login-container {
178             padding: 30px 20px;
179             width: 90%;
180         }
181     }
```

Back End - API Integration

API Endpoint (main.py)
The file below is the server side API endpoint. It waits for a POST request on the endpoint /hi. Afterwards, it responds by calling the generateImage() which constructs the neural network and loads the weight from the file generator.h5 to generate an image. Furthermore, the generated image is encoded into a Base64 string and then formatted into a URL. Finally the URL is packaged into a JSON file and sent back to the client to, straight away, use the URL to display the image.

Data sent

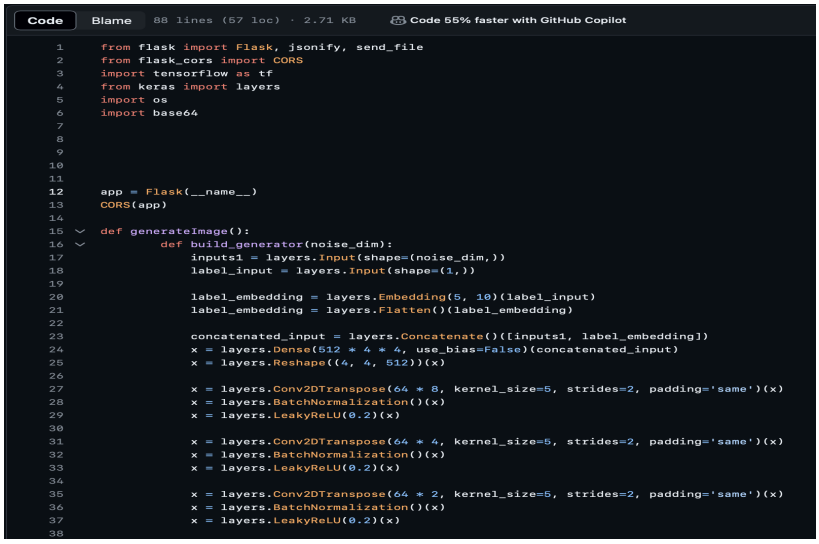
- Client side POST request.

Flask API

- Server-side receives the prompt. generateImage() is called to construct a neural network, load in the weights and generate an image then package it and send it back.

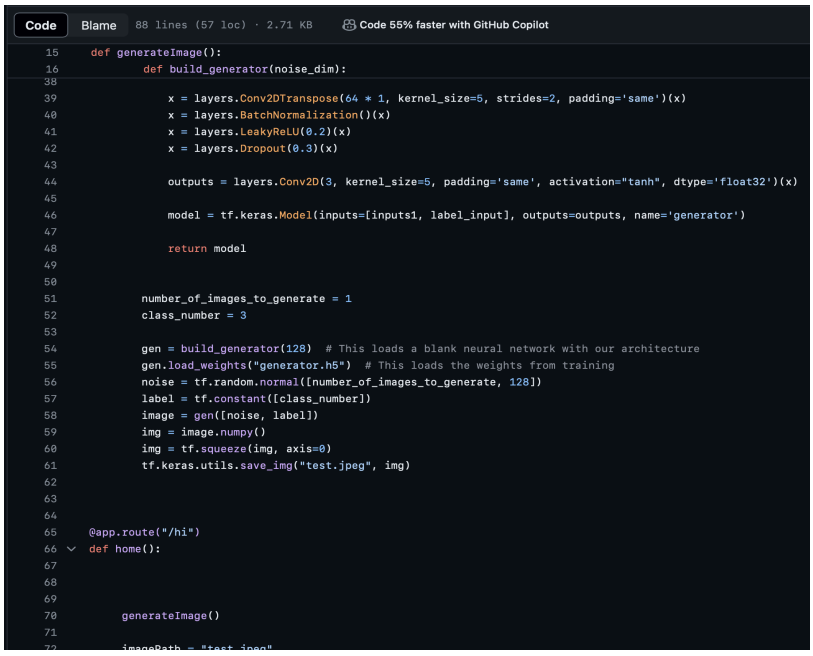
Output

- Generated image is sent back and is directly displayed because of how the image is formatted as a data URL.



Code Blame 88 lines (57 loc) · 2.71 KB Code 55% faster with GitHub Copilot

```python
from flask import Flask, jsonify, send_file
from flask_cors import CORS
import tensorflow as tf
from keras import layers
import os
import base64




app = Flask(__name__)
CORS(app)

def generateImage():
        def build_generator(noise_dim):
            inputs1 = layers.Input(shape=(noise_dim,))
            label_input = layers.Input(shape=(1,))

            label_embedding = layers.Embedding(5, 10)(label_input)
            label_embedding = layers.Flatten()(label_embedding)

            concatenated_input = layers.Concatenate()([inputs1, label_embedding])
            x = layers.Dense(512 * 4 * 4, use_bias=False)(concatenated_input)
            x = layers.Reshape((4, 4, 512))(x)

            x = layers.Conv2DTranspose(64 * 8, kernel_size=5, strides=2, padding='same')(x)
            x = layers.BatchNormalization()(x)
            x = layers.LeakyReLU(0.2)(x)

            x = layers.Conv2DTranspose(64 * 4, kernel_size=5, strides=2, padding='same')(x)
            x = layers.BatchNormalization()(x)
            x = layers.LeakyReLU(0.2)(x)

            x = layers.Conv2DTranspose(64 * 2, kernel_size=5, strides=2, padding='same')(x)
            x = layers.BatchNormalization()(x)
            x = layers.LeakyReLU(0.2)(x)

```



Code Blame 88 lines (57 loc) · 2.71 KB Code 55% faster with GitHub Copilot

```python
def generateImage():
        def build_generator(noise_dim):

            x = layers.Conv2DTranspose(64 * 1, kernel_size=5, strides=2, padding='same')(x)
            x = layers.BatchNormalization()(x)
            x = layers.LeakyReLU(0.2)(x)
            x = layers.Dropout(0.3)(x)

            outputs = layers.Conv2D(3, kernel_size=5, padding='same', activation="tanh", dtype='float32')(x)

            model = tf.keras.Model(inputs=[inputs1, label_input], outputs=outputs, name='generator')

            return model


        number_of_images_to_generate = 1
        class_number = 3

        gen = build_generator(128)  # This loads a blank neural network with our architecture
        gen.load_weights("generator.h5")  # This loads the weights from training
        noise = tf.random.normal([number_of_images_to_generate, 128])
        label = tf.constant([class_number])
        image = gen([noise, label])
        img = image.numpy()
        img = tf.squeeze(img, axis=0)
        tf.keras.utils.save_img("test.jpeg", img)



@app.route("/hi")
def home():



    generateImage()

    imagePath = "test.jpeg"
```

```python
@app.route("/hi")
def home():



    generateImage()

    imagePath = "test.jpeg"
    try:
        with open(imagePath, "rb") as imageFile:
          imageEncoded = base64.b64encode(imageFile.read()).decode()
          imageData = f"{imageEncoded}"
          return jsonify({"image": imageData})
    except FileNotFoundError:
          return jsonify({"error": "Failed to load image"}), 500

    # return send_file('test.jpeg')



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

File required to run AI model on Render (requirements.txt)

The file below is required to run the Flask file incorporating the AI model on Render. The requirements.txt has all the dependencies required to run main.py listed for Render to install. Since main.py has the AI model code and is generating images to be displayed on a web application it requires the latest versions of the relevant libraries. The main libraries used are TensorFlow for AI model, Gunicorn for web development with python, Flask for python web development, and absl-py for the utility it provides to TensorFlow.

Features

- List libraries needed to run the application

Code Blame 4 lines (4 loc) · 64 Bytes

```
absl-py==0.12.0
tensorflow==2.6.0
flask==2.0.1
gunicorn==20.1.0
```