Internal Documentation

REMEMBER TO USE REACT FOR THE FRONTEND AND PYTHON FOR THE BACKEND

Running the program

Run the program frontend with npm start in the frontend folder, the client would see this as a website to connect to via a URL

Run the program backend by running mergedApp.py in 4p02 Testing and then the postpioneer folder.

The video capability is in testing, it is in the videodiffusion.py, currently need someone with an NVIDIA GPU to be able to test it before it can be integrated.

Model

We use DeepSeek R1 for text generation and image prompt generation with the 1.5b version for testing purposes. Depending on hardware during a demonstration, one can up the version parameter count to at least 7b with a recommended 8b and if possible 14b. One can choose to go beyond that but most of us do not have the hardware for that.

We also use Stable Diffusion for image generation, it is the most compatible in my experience but if you find something better, try it out.

This can all be changed here with these two parameters.

```
MODEL_ID = "CompVis/stable-diffusion-v1-4"
CURRENT_MODEL = "deepseek-r1:1.5b"
```

If an NVIDIA GPU is present uncomment the top two lines and comment out the bottom two lines for a large performance increase (as it is currently set to run on CPU as our team has varying hardware specifications)

```
#pipe = StableDiffusionPipeline.from_pretrained(MODEL_ID, torch_dtype=torch.float16)
#pipe.to("cuda")
pipe = StableDiffusionPipeline.from_pretrained(MODEL_ID, torch_dtype=torch.float32)
pipe.to("cpu")
```

When working with DeepSeek, it provides thinking information in the output, remove it using something like this for simple text (if the text is user facing sanitize the extra newlines if necessary at the top of the message):

```
return response['message']['content'].split("</think>",1)[1]
```

When using the image generation reduce the step count to anything you want to speed up the development process but make sure to have it be 20+ for product demos or shipping to release (I believe that 20-30 works best)

```
image = pipe(prompt, num_inference_steps=20).images[0]
```

Image Transfer

To transfer an image between the backend and the frontend, convert it to Base64.

```
image = pipe(prompt, num_inference_steps=20).images[0]
buffered = BytesIO()
image.save(buffered, format="PNG")
return base64.b64encode(buffered.getvalue()).decode("utf-8")
```

Then take this Base64 and pass it in a message to the frontend

```
base64_image = generatepostImage(tone,topic)
return jsonify(
    {"message": f"{data}", "image": f"{base64_image}", "postID": postID}), 200
```

The frontend can take this Base64 and accept it in the image tag:

```
const [image, setImage] = useState("");
```

```
{image && (
      <div style={{ textAlign: "center", marginTop: "20px" }}>
        <h3>Generated Image:</h3>
        <img src={image} alt="Generated" style={{ width: "100%", maxHeight: "400px", borderRadius: "8px" }} />
      </div>
    )}
```

Prompt Generation

Currently our prompt for text generation looks like this. Edit to your heart's desire, when updated please remember to update this document:

```
if scraped_text is not None:
    initial_prompt = f"Write a {tone} social media post about {topic} in the {language} language. It must be in {language}. The data is: {scraped_text}"
else:
    initial_prompt = f"Write a {tone} social media post about {topic} in the {language} language. It must be in {language}."
```

Currently our prompt for image generation looks like this. Edit to your heart's desire, when updated please remember to update this document:

```
initial_prompt = f"Write a stable diffusion prompt about {topic}, the image tone must be {tone}. Provide just the prompt!"
```

User ID Functionality

To check if a user is signed in use something like this:

```
useEffect(() => {
  firebase.auth().onAuthStateChanged(function (user) {
    if (user) {
      setUser(user);
      console.log("user already signed in:", user.email);
    } else {
      console.log("user not signed in");
    }
  });
}, []);
```

For frontend functions that need a userID to be passed to the backend:

Use this code snippet to set the userID, normally guest should not ever be reached as pages that have userID checking **SHOULD** require a user logged in check, but some special scenarios may warrant this:

```
//set userID
React.useEffect(() => {
firebase.auth().onAuthStateChanged(function(user) {
setFormData(prevData => ({
...prevData,
userid: user ? user.uid : "guest",
}));
});
},[]);
```

When initializing firebase use a code snippet like this:

```
if (!firebase.apps.length) {
  firebase.initializeApp(firebaseConfig);
}
```

Saving to Firebase (Database)

When pushing entries to the Firebase Database make a db.reference and get to the child that you want like the code snippet below, then push that data in the format you want, it should appear in the Firebase instance if done correctly.

```
ref = db.reference("Users").child(userid).child("UserPosts")
postID = ref.push({
    "topic": topic,
    "tone": tone,
    #"data": data,
    "data": "placeholder", #keep placeholder data for now so as to not fill up the database too fast
    "schedule": schedule,
    "language": language,
    "edit": edit})
```

Reading Data from Firebase (Database)

Get a reference from the database at the child that you want and use .get() to get the information regarding it.

```python
def get_linkedin_token(username):
    # Retrieve the LinkedIn token for the user
    """
    with open('users.csv', mode='r', newline='') as file:
        reader = csv.reader(file)
        for row in reader:
            if row[0] == username:
                return row[2] if len(row) > 2 and row[2] else None
    return None
    """
    return db.reference("Users").child(username).child("Credentials").child('LinkedIn').get()
```

Here is how it would be used in this example:

```python
if platform == 'linkedin':
    access_token = get_linkedin_token(session['username'])
    if not access_token:
        flash('User not authenticated with LinkedIn.')
        return redirect(url_for('settings'))
    try:
        result = make_linkedin_post(access_token, generated_post)
        flash('Post created successfully on LinkedIn!')
    except Exception as e:
        flash(f"Error posting to LinkedIn: {str(e)}")
```

Scheduling

Try to keep scheduler start at the bottom of the code, just before the app.run

```python
# Start the scheduler
scheduler.start()
```

The code works by triggering every hour and comparing to the current system time to see if it is appropriate to trigger a generate post for the hour, daily, weekly, biweekly, and monthly times. **It needs to post the image to an endpoint but this is currently not being added so as to not clog the database, add this in the final product.**

```python
@scheduler.task('cron', id='job_2', hour='*', minute='0')
def hourly_trigger():
    print("This job runs every hour!")
    users_ref = db.reference("Users")
    users = users_ref.get()
    # Iterate and print each user and their posts
    current_time = datetime.datetime.now()
    if users:
        for user_id, user_data in users.items():
            print(f"User ID: {user_id}")
            if "UserPosts" in user_data:
                for post_id, post_data in user_data["UserPosts"].items():
                    if "schedule" in post_data:
                        if post_data["schedule"] == "hourly":
                            print("Hourly post")
                            print(generatePostText(post_data["tone"], post_data["topic"], post_data["language"]))
                            generatepostImage(post_data["tone"], post_data["topic"])
                            print("image generated")
                        elif (post_data["schedule"] == "daily") and (current_time.hour == 0):
                            print("Daily post")
                            print(generatePostText(post_data["tone"], post_data["topic"], post_data["language"]))
                            generatepostImage(post_data["tone"], post_data["topic"])
                            print("image generated")
                        elif (post_data["schedule"] == "weekly") and (current_time.weekday() == 0) and (current_time.hour == 0):
                            print("Weekly post")
                            print(generatePostText(post_data["tone"], post_data["topic"], post_data["language"]))
                            generatepostImage(post_data["tone"], post_data["topic"])
                            print("image generated")
                        elif (post_data["schedule"] == "biweekly") and (current_time.weekday() == 0) and (current_time.hour == 0) and (current_time.day % 14 == 0):
                            print("Biweekly post")
                            print(generatePostText(post_data["tone"], post_data["topic"], post_data["language"]))
                            generatepostImage(post_data["tone"], post_data["topic"])
                            print("image generated")
                        elif (post_data["schedule"] == "monthly") and (current_time.day== 1) and (current_time.hour == 0):
                            print("Monthly post")
                            print(generatePostText(post_data["tone"], post_data["topic"], post_data["language"]))
                            generatepostImage(post_data["tone"], post_data["topic"])
                            print("image generated")
            else:
                print("  No UserPosts found.")
    else:
        print("No users found in the database.")
```