**Code for the Application**

* For the project folder:

api\_integration\_app/

│

├── app.py # Main application file

├── templates/ # Folder for HTML templates

│ ├── index.html # Home page template

│ └── map.html # Map display page template

├── .gitignore # Git ignore file (to avoid uploading sensitive data)

└── requirements.txt # File for required dependencies

* Main Phyton File

import tweepy

import sendgrid

from sendgrid.helpers.mail import Mail, Email, To, Content

from flask import Flask, render\_template, request, redirect, url\_for

import requests

# Initialize Flask app

app = Flask(\_\_name\_\_)

# Set your API keys and secrets

TWITTER\_API\_KEY = 'your\_twitter\_api\_key'

TWITTER\_API\_SECRET\_KEY = 'your\_twitter\_api\_secret\_key'

TWITTER\_ACCESS\_TOKEN = 'your\_twitter\_access\_token'

TWITTER\_ACCESS\_TOKEN\_SECRET = 'your\_twitter\_access\_token\_secret'

GOOGLE\_MAPS\_API\_KEY = 'your\_google\_maps\_api\_key'

SENDGRID\_API\_KEY = 'your\_sendgrid\_api\_key'

# Twitter Authentication

auth = tweepy.OAuth1UserHandler(

consumer\_key=TWITTER\_API\_KEY,

consumer\_secret=TWITTER\_API\_SECRET\_KEY,

access\_token=TWITTER\_ACCESS\_TOKEN,

access\_token\_secret=TWITTER\_ACCESS\_TOKEN\_SECRET

)

twitter\_api = tweepy.API(auth)

# SendGrid Client

sg = sendgrid.SendGridAPIClient(api\_key=SENDGRID\_API\_KEY)

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/post\_tweet', methods=['POST'])

def post\_tweet():

tweet = request.form['tweet']

try:

# Post a tweet

twitter\_api.update\_status(tweet)

return redirect(url\_for('home'))

except tweepy.TweepError as e:

return f"Error: {e}"

@app.route('/show\_map', methods=['GET'])

def show\_map():

location = request.args.get('location', '')

map\_url = f"https://maps.googleapis.com/maps/api/staticmap?center={location}&zoom=14&size=400x400&key={GOOGLE\_MAPS\_API\_KEY}"

return render\_template('map.html', map\_url=map\_url)

@app.route('/send\_email', methods=['POST'])

def send\_email():

email\_content = request.form['email\_content']

from\_email = Email("your-email@example.com")

to\_email = To("recipient@example.com")

subject = "Test Email from SendGrid"

content = Content("text/plain", email\_content)

mail = Mail(from\_email, to\_email, subject, content)

try:

response = sg.send(mail)

if response.status\_code == 202:

return redirect(url\_for('home'))

return f"Failed to send email. Status Code: {response.status\_code}"

except Exception as e:

return str(e)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

* Home Page

<!DOCTYPE html>

<html>

<head>

<title>API Integration App</title>

</head>

<body>

<h1>API Integration App</h1>

<form action="/post\_tweet" method="POST">

<input type="text" name="tweet" placeholder="Enter tweet" required>

<button type="submit">Post Tweet</button>

</form>

<form action="/send\_email" method="POST">

<textarea name="email\_content" placeholder="Enter email content" required></textarea>

<button type="submit">Send Email</button>

</form>

<form action="/show\_map" method="GET">

<input type="text" name="location" placeholder="Enter location" required>

<button type="submit">Show Map</button>

</form>

</body>

</html>

* Map Display

<!DOCTYPE html>

<html>

<head>

<title>Map Display</title>

</head>

<body>

<h1>Map</h1>

<img src="{{ map\_url }}" alt="Map">

<br><br>

<a href="/">Back to Home</a>

</body>

</html>

* Git Ignore File

# Python bytecode

\_\_pycache\_\_/

\*.pyc

\*.pyo

# Virtual environment

venv/

.env/

# API Keys (Add the path to a .env or other files that store sensitive keys)

\*.env

# IDE files

.vscode/

.idea/

* Dependencies

Flask==2.1.1

tweepy==4.9.0

sendgrid==6.9.1

requests==2.26.0

To Run the Operation

git clone https://github.com/your\_username/api\_integration\_app.git

cd api\_integration\_app

pip install -r requirements.txt

python app.py

**Web Browser Access:** [**http://127.0.0.1:5000/**](http://127.0.0.1:5000/)

**Written Report: API Integration Web Application**

In today’s rapidly evolving technological landscape, web applications have become powerful tools for users to interact with various services and retrieve valuable information. One of the key aspects of modern web development is the integration of third-party Application Programming Interfaces (APIs) to extend the functionality of applications. APIs allow developers to access external services without having to build complex systems from scratch. The goal of this project was to develop a **web application that integrates three popular APIs**: **Twitter API**, **Google Maps API**, and **SendGrid API**, using the Flask web framework. By integrating these APIs, the application allows users to perform tasks such as posting tweets to Twitter, displaying geographic locations on a map, and sending emails through the SendGrid service. This report will explore the purpose of the project, the features of the application, the technologies used, and a detailed explanation of how each of the integrated APIs was implemented.

**Purpose of the Project**

The primary objective of the project was to demonstrate how multiple third-party APIs can be integrated into a single web application, providing a seamless experience for the user. The application aims to leverage the power of social media, location-based services, and email services to enhance the capabilities of a simple Flask app. By utilizing these three APIs, the project addresses the real-world need for integrating different types of services into a cohesive user experience. For instance, users can interact with their Twitter account, explore locations using Google Maps, and send emails to contacts—all within a single application. Each API integration not only adds useful functionality to the web app but also provides an opportunity to learn about API authentication, managing API keys, and interacting with external services using RESTful APIs.

**Technologies Used**

The project is built using **Flask**, a lightweight Python web framework known for its simplicity and flexibility in handling web applications. Flask is ideal for projects that require rapid development with minimal configuration. The app follows the Model-View-Controller (MVC) architecture, which helps separate the logic of the application from the user interface, allowing for better maintainability and scalability.

For the API integrations, the following services were used:

* **Twitter API**: The application integrates with the Twitter API to enable users to post tweets from the web interface. OAuth is used for user authentication, ensuring secure access to the user's Twitter account.
* **Google Maps API**: Users can enter a location, and the app generates a static map showing the location on a map using the Google Maps API. This feature allows users to visualize geographic data.
* **SendGrid API**: The SendGrid API is used to send emails from the app. Users can input email content and send it to any recipient, which makes it easy to integrate email communication features into web applications.

These APIs not only serve as the foundation of the application’s functionality but also exemplify how modern web development increasingly relies on external services to provide rich, user-friendly experiences.

**Detailed Explanation of Features**

The web application consists of several features, each corresponding to one of the integrated APIs. These features are designed to be simple yet functional, allowing users to interact with them through a user-friendly interface.

**1. Twitter API Integration**

The first feature of the application allows users to post tweets to Twitter directly from the web interface. This functionality is powered by the **Twitter API**, which provides programmatic access to Twitter’s platform. To interact with the Twitter API, the app uses **OAuth 1.0** for user authentication. OAuth allows users to securely grant the application access to their Twitter account without needing to share their password. Instead, the user authenticates by granting permission to the application through the Twitter interface, which then returns an access token that the app uses to post tweets on the user’s behalf.

When a user submits a tweet, the Flask app sends a request to the Twitter API with the tweet content. The API responds by posting the tweet to the user’s Twitter feed. This feature is particularly useful for automating social media interactions or integrating social sharing within other applications. By allowing users to post directly from the app, the integration enhances the user experience and provides a practical use case for API-driven development.

**2. Google Maps API Integration**

The second feature of the application utilizes the **Google Maps API** to display a map based on a location input by the user. The app allows users to type in an address or place, and Google Maps returns a static map image centered on that location. The map image is then embedded in the web page for the user to view.

To achieve this functionality, the application constructs a URL that includes the desired location and the required API key. The URL points to the **Google Maps Static API**, which generates a static image of the map that can be embedded in the page. The app uses a simple GET request to fetch this image, and it dynamically updates the map based on the user’s input.

The Google Maps API is invaluable in applications that require geolocation or map-based services. By integrating it into the application, users can get a visual representation of a location without needing to interact with complex interactive maps. This feature could be expanded further to include more interactive map features, such as placing markers or fetching nearby points of interest.

**3. SendGrid API Integration**

The third key feature of the application is the ability to send emails using the **SendGrid API**. Email is a vital communication tool, and integrating email functionality into a web application is a common requirement for many services. Using the SendGrid API, the app allows users to send emails by composing the content in a simple text field and specifying the recipient's email address.

The integration with SendGrid is handled by making POST requests to the SendGrid email service API, passing the email content, subject, sender’s email address, and recipient’s email address. The API sends the email via SendGrid’s email delivery infrastructure. This eliminates the need for setting up an SMTP server or dealing with the complexities of email sending, as SendGrid handles deliverability, tracking, and error management.

The SendGrid integration is especially useful for applications that need to send transactional emails, such as registration confirmations, password reset instructions, or promotional content. By integrating SendGrid, the application provides a reliable and scalable solution for email communication.

**User Experience and Application Flow**

The application is designed to be user-friendly and intuitive. Upon visiting the homepage, users are greeted with three distinct sections, each corresponding to one of the API integrations. The user interface consists of simple form fields for each feature:

* A text input for composing a tweet.
* A text field for entering a location.
* A text area for composing an email.

Once a user fills in one of the forms and submits it, the app processes the input and interacts with the corresponding API to execute the desired action. For the Twitter feature, a tweet is posted; for Google Maps, a static map is displayed; and for SendGrid, an email is sent. Each action is processed asynchronously, with the user being redirected to a success page or receiving an error message if something goes wrong.

**Conclusion**

The **API Integration Web Application** project demonstrates how multiple external APIs can be integrated into a single web application to enhance its functionality. By leveraging **Twitter API**, **Google Maps API**, and **SendGrid API**, the application provides valuable services such as posting tweets, displaying geographic locations, and sending emails. This project not only showcases the practical use of APIs but also highlights the importance of modern web development practices, such as handling API authentication, interacting with RESTful services, and ensuring a seamless user experience. As APIs continue to play a pivotal role in application development, this project serves as a stepping stone for learning how to integrate and utilize third-party services in web applications.