## **Installation**

Pip install Flask

Pip install pywin32

Navigate to the Python(ex: C:\Program Files\Python312) directory

python Scripts/pywin32\_postinstall.py -install

## **Create Flask application** Create two python files test.py

## from flask import Flask, request, jsonify

## Import sqlite3

## import logging

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

## # Function to initialize the database

## def init\_db():

## conn = sqlite3.connect('employees.db')

## c = conn.cursor()

## c.execute('''CREATE TABLE IF NOT EXISTS employees

## (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, position TEXT)''')

## conn.commit()

## conn.close()

## # Initialize the database

## init\_db()

## # Endpoint to add an employee

## @app.route('/add\_employee', methods=['POST'])

## def add\_employee():

## logging.info('This is add\_employee API')

## try:

## data = request.json

## name = data['name']

## position = data['position']

## 

## conn = sqlite3.connect('employees.db')

## c = conn.cursor()

## c.execute("INSERT INTO employees (name, position) VALUES (?, ?)", (name, position))

## conn.commit()

## conn.close()

## return jsonify({'message': 'Employee added successfully'}), 201

## except Exception as e:

## return jsonify({'error': str(e)}), 400

## @app.route('/employees', methods=['GET'])

## def get\_employees():

## logging.info('This is employees API')

## try:

## conn = sqlite3.connect('employees.db')

## c = conn.cursor()

## c.execute("SELECT \* FROM employees")

## employees = c.fetchall()

## conn.close()

## employee\_list = []

## for employee in employees:

## employee\_dict = {'id': employee[0], 'name': employee[1], 'position': employee[2]}

## employee\_list.append(employee\_dict)

## return jsonify(employee\_list), 200

## except Exception as e:

## return jsonify({'error': str(e)}), 500

## @app.route('/api/hello')

## def hello():

## logging.info('This is hello API')

## return jsonify({'message': 'Hello, World!'})

## if \_\_name\_\_ == '\_\_main\_\_':

## logging.basicConfig(filename='app.log',level=logging.INFO)

## app.run(host='0.0.0.0', port=5000)

## app\_service.py

import time

import win32serviceutil

import win32service

import win32event

import servicemanager

import socket

import sys

from flask import request

from test import app

class FlaskAppService:

"""Silly little application stub"""

def stop(self):

"""Stop the service"""

self.running = False

def run(self):

"""Main service loop. This is where work is done!"""

self.running = True

app.run(host='0.0.0.0', port=5000)

while self.running:

time.sleep(10) # Important work

servicemanager.LogInfoMsg("Service running...")

class AppServerSvc(win32serviceutil.ServiceFramework):

\_svc\_name\_ = "FlaskAppService"

\_svc\_display\_name\_ = "Flask App Service"

\_svc\_description\_ = "This service runs a Flask app."

def \_\_init\_\_(self, args):

win32serviceutil.ServiceFramework.\_\_init\_\_(self, args)

self.timeout = 30000

self.hWaitStop = win32event.CreateEvent(None, 0, 0, None)

socket.setdefaulttimeout(60)

def SvcStop(self):

self.ReportServiceStatus(win32service.SERVICE\_STOP\_PENDING)

self.service\_impl.stop()

self.ReportServiceStatus(win32service.SERVICE\_STOPPED)

def SvcDoRun(self):

self.ReportServiceStatus(win32service.SERVICE\_START\_PENDING)

self.service\_impl = FlaskAppService()

self.ReportServiceStatus(win32service.SERVICE\_RUNNING)

self.service\_impl.run()

if \_\_name\_\_ == '\_\_main\_\_':

if len(sys.argv) == 1:

servicemanager.Initialize()

servicemanager.PrepareToHostSingle(AppServerSvc)

servicemanager.StartServiceCtrlDispatcher()

else:

win32serviceutil.HandleCommandLine(AppServerSvc)

**Running the Service:**

Open Command Prompt as Administrator.

Navigate to the project directory

Install the service

python app\_service install  
  
Start the service

python app\_service start

**Accessing the API:**

<http://localhost:5000/api/hello>

http://localhost:5000/add\_employee

Stop the service

python app\_service stop

## 