# CHEM1AL Automation System

## Overview

This Python project automates various administrative tasks for the CHEM1AL course at UC Berkeley. It's designed to help Graduate Student Instructors (GSIs) manage student inquiries, attendance, lab reports, and email communications more efficiently.

## Features

- Processes student contact form submissions
- Manages lab rescheduling and online lab requests
- Handles extension requests for lab reports
- Handles requests to reopen pre-lab quizzes
- Checks student attendance and lab report submissions
- Integrates with Google Sheets, Gmail, and OpenAI's GPT for various functionalities

## Prerequisites

Before you begin, ensure you have met the following requirements:

- Install your preferred development environment
- Python 3.x installed
- Access to the course's Google Sheets and Gmail account
- OpenAI API key for GPT integration

## Main Components

- `CHEM1AL.py`: The main script containing all the automation logic
- `CHEM1AL_Dicts.py`: Contains dictionaries with course-specific information
- `timestamp.txt`: Stores the timestamp of the last processed contact form entry
- Credentials for API keys
    - `chem-1al-creds.json`
    - `gmail_creds.json`
    - `token.json`
- Scripts for authenticating the google API keys
    - `googleApiQuickstart.py`
    - `googleApis.py`

# Usage

The script can be run with different command-line arguments to perform various tasks:

1. Process student contact form submissions and populates the Gmail draft folder

   Resources needed: Weekly checklist sheet, Student response sheet, Gmail

```
Unset
python CHEM1AL.py You write my emails
```

2. Use ChatGPT to draft responses to student emails. (Not functional in current version)

   Resources needed: Gmail, chatGPT

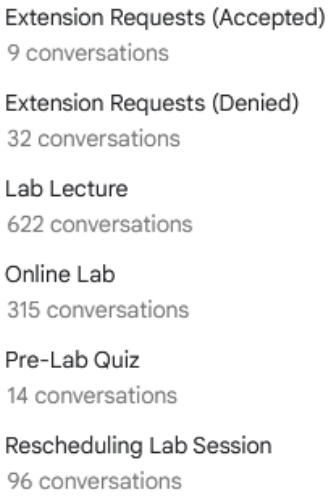
```
Unset
python CHEM1AL.py You are a chatGPT wrapper
```

3. Sort sent emails into appropriate Gmail folders.

   Resources needed: Gmail

```
Unset
python CHEM1AL.py You sort my emails
```

In order for this to work without a rewrite set up your folders with at least the labels shown below



4. Check student attendance. Students with ≥3 absences are returned to the terminal

    Resources needed: Weekly checklist sheet

```
Unset
python CHEM1AL.py You check attendance
```

5. Count submitted lab reports. Students with ≥3 missing submissions are returned to the terminal

    Resources needed: Weekly checklist sheet

```
Unset
python CHEM1AL.py You count lab reports
```

# Maintenance

- Update the `CHEM1AL_Dicts.py` file with any changes in course structure, GSI assignments, or lab schedules.
- Monitor the OpenAI API usage and ensure the API key remains valid.
- Periodically check the Google Sheets and Gmail API access to ensure the credentials haven't expired.

# Troubleshooting

If you encounter issues:

1. Check that all credential files are present and valid
2. Ensure the Google Sheets and contact form structures haven't changed
3. Verify that the OpenAI API key is correctly set up
4. Check the console output for any error messages

Note: I haven’t included chatGPT API instructions because since writing this their API has updated which has broken the chatGPT wrapper function. If you would like to update that you’ll have to create your own chatGPT account and read their documentation.

For any questions email nickciolkowski@berkeley.edu

# Setup

1. Download this repository to your local machine.
2. Install the required Python packages:

```
Python
pip install gspread oauth2client google-auth-oauthlib google-auth-httplib2
google-api-python-client openai
```

3. Set up credentials (See Google API Setup):
    a. Check that `chem-1al-creds.json` is in the project root for Google Sheets access
    b. Check that `gmail_creds.json` is in the project root for Gmail API access
    c. Set up your OpenAI API key (not required)
4. Update the `CHEM1AL_Dicts.py` file with the current semester's information:
    a. Section times

```
Python
sectionTimes = {
    "T 8-11": ("201", "202", "203"),
    "T 1-4": ("211", "212", "213", "214"),
    "W 1-4": ("311", "312", "313", "314"),
    "W 6-9": ("321", "322"),
    "Th 1-4": ("411", "412", "413", "414"),
    "F 1-4": ("511")}
```

    b. GSI Sections

```
Python
gsiSections = {
    "Drew Salmon": ("201", "211"),
    "Andres Arraiz": ("202", "213"),
    "Diego Novoa": ("203", "312"),
    "Sunnie Kong": ("212", "412"),
    "Utkarsh Tiwari": ("313", "411"),
```

```
    "Aaron Guo": ("214", "322"),
    "Jack Feldner": ("314", "413"),
    "Anna Kurianowicz": ("321", "511"),
    "Henry Phan": ("311", "414")}
```

c. GSI Emails

Python

```
gsiEmails = {
    "Drew Salmon": 'example@berkeley.edu',
    "Andres Arraiz": 'example@berkeley.edu',
    "Diego Novoa": 'example@berkeley.edu',
    "Sunnie Kong": 'example@berkeley.edu',
    "Utkarsh Tiwari": 'example@berkeley.edu',
    "Aaron Guo": 'example@berkeley.edu',
    "Jack Feldner": 'example@berkeley.edu',
    "Anna Kurianowicz": 'example@berkeley.edu',
    "Henry Phan": 'example@berkeley.edu',}
```

d. Lab Dates

Python

```
labDates = {
    'Lab1': datetime(2024, 1, 27),
    'Lab2': datetime(2024, 2, 3),
    'Lab3': datetime(2024, 2, 10),
    'Lab4': datetime(2024, 2, 17),
    'Lab5': datetime(2024, 2, 24),
    'Lab6': datetime(2024, 3, 2),
    'Lab7': datetime(2024, 3, 9),
```

```
    'Lab8': datetime(2024, 3, 16),
    'Lab9': datetime(2024, 3, 23),
    'Lab10': datetime(2024, 4, 6),
    'Lab11': datetime(2024, 4, 13),
    'LabFinal': datetime(2025, 1, 1)}   #This date is pushed further back otherwise the attendance counter breaks at the end of the semester
```

5. Update the `CHEM1AL.py` file and redefine the headGsiName variable.

Python
```
headGsiName = "Your Name"
```

6. Update the `CHEM1AL.py` file with the google sheets that need to be accessed.

Python
```
#Sheet plugins

sheetContactForm = client.open("Spring 24 CHEM1AL Contact Form")
#Student contact sheet input
seqResponse = sheetContactForm.get_worksheet(1)
#Sequential fix
sheetAttendance = client.open("1AL Weekly Checklist Spring 2024")
#Student Attendance info
print("Google sheets API implemented successfully")
```

7. If needed, update the `CHEM1AL.py` file email templates to fit new semester requirements.

# Google API Setup

To use the Google Sheets and Gmail functionalities, you'll need to set up Google Cloud Platform and enable the necessary APIs. Follow these steps:

1. Log into the Google Cloud Project:
    - Go to the Google Cloud Console
    - Click on "Select a project" at the top of the page
    - Click "CHEM 1AL"
2. Create credentials for Gmail:
    - Go back to "APIs & Services" > "Credentials"
    - Click "Create Credentials" and select "OAuth client ID"
    - Choose "Desktop app" as the application type
    - Click "Create" and download the JSON file
    - Delete the old `gmail_creds.json` in the project directory
    - Rename the new downloaded file to `gmail_creds.json` and place it in your project directory.
    - Delete the existing `token.json` in the project directory
    - Execute the `googleApiQuickstart.py` script to authenticate with the new Gmail account and generate a new token file.
    - This will open a browser window for you to log in to the new Gmail account and grant permissions. Follow the instructions in the browser.
    - After successful authentication, a new `token.json` file will be created in your project directory. This file stores the access and refresh tokens
3. Share the Google Sheets:
    - Open the Google Sheets you want to access
    - Click the "Share" button
    - In the "Add people and groups" field, paste the client_email from your `chem-1al-creds.json` file
    - Make sure to give it "Editor" access
    - Click "Share"
    - Update the `CHEM1AL.py` file with the new sheet names
4. Authorize Gmail access:
    - The first time you run the script that uses Gmail, it will open a browser window asking you to authorize the application
    - Follow the prompts to grant access to the appropriate Gmail account