**Assignment Readme**

**Requirements Engineering:**

1. **Email Retrieval :**

* **Setting Up Gmail API:**

**Go to the Google Developer Console.**

**Create a new project.**

**Enable the Gmail API for that project.**

**Create OAuth 2.0 client credentials, and download the JSON file (client\_secret.json) to your credentials/ directory.**

* **Implement the code to interact with the Gmail API**
* **To retrieve emails from Gmail using the Gmail API, follow these steps:**

**. Set up a project in the Google Developer Console and enable the Gmail API.**

**. Generate credentials for your application (OAuth 2.0 client ID) and download the JSON file.**

**. Use the Gmail API to list emails with a specific subject line and download attachments.**

* **Download the pdf attached to the mail with mail subject ad “Bank Statement”**

1. **PDF Parsing**
2. **Search Transaction**
3. **Get Balance**

**Implementation Details:**

**Bank Statement Parser Application**

**This Python application parses and extracts transaction information from PDF bank statements received via Gmail. It uses Flask for the user interface and interacts with the Gmail API for fetching emails and attachments.**

**Prerequisites:**

1. **Gmail API Credentials: Set up Gmail API credentials and provide the client\_secret.json file for OAuth 2.0 authorization.**
2. **Python Libraries: Ensure the required libraries are installed using pip install:**

**Copy code**

***flask pdfplumber google-auth-oauthlib google-api-python-client google-auth-httplib2 google-auth-httplib2***

**Configuration and Authorization:**

1. **Gmail API credentials are configured using OAuth2.0 authorization. The CLIENT\_SECRET\_FILE and TOKEN\_PICKLE\_FILE specify client secrets and token storage.**
2. **The get\_credentials function authorizes the application and obtains Gmail API credentials.**
3. **The Gmail API is initialized using obtained credentials.**

**Functions:**

1. **get\_emails\_with\_subject(subject): Retrieves email messages with a specified subject from the user's Gmail account using the Gmail API.**
2. **download\_attachments(email\_id): Downloads PDF attachments from emails, stores them locally in the pdf\_files directory, and uses the Gmail API to access attachments.**
3. **parse\_and\_retrieve(): Parses PDF files within the pdf\_files directory using pdfplumber to extract transaction data. Extracted transactions (date, description, and amount) are stored in a list.**

**Routes and Endpoints:**

1. **/list\_transactions: Parses PDF files, extracts transactions, and renders an HTML template with the list of transactions.**
2. **/search\_transactions: Accepts start and end dates, searches for transactions within the specified range, and returns them as a JSON response. Includes date parsing and filtering exception handling.**
3. **/get\_balance: Calculates the account balance as of a specific date and returns it as a JSON response. Handles date parsing and balance calculation exceptions.**
4. **/gmail\_service: Retrieves emails with the subject "Bank Statement" from Gmail, downloads PDF attachments, parses transactions, and lists them. Renders an HTML template indicating the number of processed emails.**
5. **/: The default home route renders an HTML template for the home page of the application.**

**Running the Application:**

**Running the application starts a Flask server. Access the user interface via the provided URL in a web browser.**

**Exception Handling:**

**The code includes exception handling for potential issues like date parsing errors and Gmail API errors. Errors are logged, and appropriate responses are returned to the user.**

**Note:**

* **Serve HTML templates from the templates directory.**
* **Ensure the code and templates align with your specific use case and requirements.**

**How to Use:**

1. **Set up Gmail API credentials and specify the client secret file.**
2. **Install the required Python libraries.**
3. **Run the application.**
4. **Access the web interface to retrieve, search, and calculate transactions from Gmail and parsed PDFs.**

**Note: The attached PDF should be in the format provided as Bank\_Statement\_format. By running the app for the first time you need to provide authorization which then create token.pickle .You need to store your credential.JSON in credit folder. You need to create pdf\_files folder to store downloaded pdfs'.**

**Result:**

