Gloat Integrations Home Assignment

Overview:
We have 3 types of HCM (Human Capital Management) integrations with our customers. The integration's purpose is to receive which users are whitelisted to go into the platform.
Gloat's users fields, to be used throughout the 3 integrations are listed at the bottom of this page as are field mapping instructions and additional essential information for this assignment.

We should be able to execute each one of the implemented integrations below at any given time (using python/flask/bash/etc..)

Integration Types:

1. We have "Users feed" (csv file) which is sent daily via SFTP. Implement an integration process for this feed, according to the following guidelines

File name: **auth-users-<today's-date>.csv**

1.1. Check if file arrived (decide how to periodically check if file exists)
1.2. Process file (only once)
1.3. Map client fields into "Gloat" fields - Note: not all fields might map "nicely" but we want to keep all data coming from CSV.
1.4. Handle pre-processing errors (invalid file, etc.) and handle errors in general (missing fields, wrong fields types, etc.)
1.5. Call Gloat's POST /api/v1/users/whitelist API (Technically this endpoint should already exists, but implement an empty endpoint which doesn't do anything just for the sake of this task) which whitelists & updates users (see "Gloat's whitelisting API" description below for more details)
1.6. Send a process summary report to integrations-<client name>@gloat.com

Note: this process is intended to run in a Gloat server which is dedicated for customer. (i.e the Gloat web server will be running on it as well)

For the next 2 integrations assume the fields that came in the CSV now arrive via json.

2. Now, instead of receiving the whitelisted users from a CSV file, we need to fetch the users from the client's API https://acme.com/api/v1/authorized-users?from=date&end=date

This API requires basic authenticatication using CLIENT_ID and CLIENT_SECRET and it returns list of json objects (users)

Credentials for the call:
CLIENT_ID = "BBBB"
CLIENT_SECRET = "GLOAT_INTEGRATION"
(Technically this endpoint should already exist, but implement an endpoint which returns the json written below just for the sake of this task)

2.1. Daily fetch new data from client's API
2.2. Process response and whitelists users (calls the /whitelist API)

3. Now create an endpoint that allows customers to approach Gloat and whitelist users themselves. These user feeds are "pushed" into Gloat using an API you will build for them.
   3.1. This API should require token based basic authentication
   3.2. Build a server which will serve this API
   3.3. Process request once API is called similarly to what was done in the previous 2 questions

<u>Helpful information for all integrations implementation:</u>

Gloat's user fields (to be sent to the /whitelist API):

1. user_id (int) - mandatory
2. email (string) - mandatory
3. first_name (string) - mandatory
4. last_name (string) - mandatory
5. can_access_platform (boolean)
6. manager_email (string)
7. manager_id (int)
8. department <list(string)> (all department hierarchy fields)
9. city (string) - mandatory
10. country (string)
11. business_unit (string)
12. extra_data (json)

Gloat's whitelisting API

URL: /api/v1/users/whitelist
Method: POST
Receives a list of json objects containing the keys mentioned above, whitelists the users in the list or updates the data for existing users.

Returns: 201 HTTP response upon success

Authorization details: Basic authorization
client_id = "ABCD"
client_secret = "GLOAT2020"

Field mapping guidelines:

1. Gloat department is a hierarchy field and should be comprised of all department fields coming from the feed/json (in a sorted list by hierarchy from high to low)
2. Access to platform logic is as follows: access is permitted only to users with grade level 6 and above AND positive (“Y”) value for “people leader” field
3. Any field coming from integration that doesn’t map should be saved as well in the "extra_data" field

- Keep in mind that every company has different names for each field. meaning, each company will have it's own mapping.

CSV file content

UserId,First_name,Last_name,Email_Id,Manager_userId,Manager_email,Grade level,Department level 1,Department level 2,Department level 3,Title, Division,Location,City,Country,Business_unit
A1B2C3,Mike,Ross,mike.r@gloat.com,456,Harvey.S@gloat.com,8, Human resources,SK708,HR Ops & Delivery,HR,Human resources,"Bangalore, India",Bangalore,IND,HR Manager
A2B3C4,Donna,Paulson,donna.p@gloat.com,456,Harvey.S@gloat.com,5,Human resources,SK708,HR Ops & Delivery,HR recruiter,Human resources,"Bangalore, India",Bangalore,IND,HR

Json


```
[
 {
  "UserId": "A1B2C3",
  "First_name": "Mike",
  "Last_name": "Ross",
  "Email_Id": "mike.r@gloat.com",
  "Manager_userId": 456,
  "Manager_email": "Harvey.S@gloat.com",
```

```
  "Grade level": 8,
  "Department level 1": "Human resources",
  "Department level 2": "SK708",
  "Department level 3": "HR Ops & Delivery",
  "Title": "HR",
  "Division": "Human resources",
  "Location": "Bangalore, India",
  "City": "Bangalore",
  "Country": "IND",
  "Business_unit": "HR Manager",
  "People_Leader": "Y"
 },
 {
  "UserId": "A2B3C4",
  "First_name": "Donna",
  "Last_name": "Paulson",
  "Email_Id": "donna.p@gloat.com",
  "Manager_userId": 456,
  "Manager_email": "Harvey.S@gloat.com",
  "Grade level": 5,
  "Department level 1": "Human resources",
  "Department level 2": "SK708",
  "Department level 3": "HR Ops & Delivery",
  "Title": "HR recruiter",
  "Division": "Human resources",
  "Location": "Bangalore, India",
  "City": "Bangalore",
  "Country": "IND",
  "Business_unit": "HR",
  "People_Leader": "N"
 }
]
```