Full-Stack Assignment — Pathik AI

React + Python (Flask) + PostgreSQL + Google Ads API

Objective

Build a small full-stack application that allows a user to:

1. **Create a marketing campaign** (stored locally in PostgreSQL)
2. **Publish the campaign** to a real Google Ads account using the **Google Ads API**

   Note: You should be creating Inactive campaigns(Or control by startdate)

3. Use any appropriate **campaign objective** (e.g., *Sales, Leads, Website Traffic*).
4. Preferably create a **Demand Gen Campaign**, but this is **optional** — completing a basic campaign creation is acceptable.

This assignment tests:

- API design
- React frontend
- Flask backend
- PostgreSQL integration
- Google Ads API usage
- Code quality & documentation

---

1. Backend — Python Flask

Tech Requirements

- Python 3.x
- Flask
- PostgreSQL
- SQLAlchemy (preferred)
- **Google Ads API using `GoogleAdsClient`** (This is the official library for Google Ads API requests.)

Google Ads Requirements

You will need to configure:

- Developer token
- Client ID
- Client secret
- Refresh token
- Login customer ID
- Customer account ID

(You may need to create your own Google Ads test account.)

Backend Functional Requirements

➤ 1. Create a campaign (LOCAL DB only)

Implement an API:

`POST /api/campaigns`

The API should:

- Validate request data
- Store the campaign in PostgreSQL
- Set `status = "DRAFT"` by default

Important:

You **can design the database schema yourself**. The provided example is only for reference.

Example (not mandatory):

field	type
id	UUID
name	text
objective	text
campaign_type	text
daily_budget	int
start_date	date
end_date	date
status	text
google_campaign_id	text
ad_group_name	text
ad_headline	text
ad_description	text
asset_url	text
created_at	timestamp

You are free to modify / add fields as you think appropriate.

➢ 2. Get all campaigns

`GET /api/campaigns`

➢ 3. Publish a campaign to Google Ads

`POST /api/campaigns/<id>/publish`

This should:

1. Read campaign details from local DB.
2. Initialize Google Ads client:

```
client = GoogleAdsClient.load_from_storage()
```

3. Create:
    - **Campaign**
    - **Ad Group**
    - **Ad**
    - **Asset (If needed)**
4. Preferably create a **Demand Gen campaign**, but **not mandatory**.
    - Any campaign objective (Sales, Leads, Traffic, Engagement) is allowed.
    - Candidate may choose any campaign type they are comfortable implementing.
5. Store the returned Google Campaign ID in DB.
6. Update status to `PUBLISHED`.

➢ 4. Disable Campaign

So that your account is not charged.

---

2. Frontend — React

Tech Requirements

- React
- Axios or Fetch
- Basic state management

UI Requirements

➢ Campaign Form

Form fields (recommended but flexible):

- Campaign Name
- Objective (input or select)
- Daily Budget
- Start Date / End Date
- Campaign Type (default “Demand Gen” but editable)
- Ad Group Name
- Ad Headline
- Ad Description
- Asset URL

Buttons:

- **Save Locally** → Calls `POST /api/campaigns`
- **Publish to Google Ads** → Calls `POST /api/campaigns/{id}/publish`

➢ Campaign Listing

List all saved campaigns:

- Name
- Status (DRAFT/PUBLISHED)
- Google Campaign ID
- Button: “Publish”
- Button: “Pause” (optional)

---

3. Project Expectations

✔ Clean code & folder structure

✔ README containing:

- Setup instructions
- How to run backend
- How to run frontend
- Environment variables
- Google Ads setup steps
- API documentation

✔ Proper API error handling

✔ Validation on inputs

✔ Simple UI but functional

---

4. Optional Bonus

- Docker / Docker Compose
- Form validation (Yup, Formik)
- Logging on backend
- Unit tests
- Redux or Zustand state management

---

5. Submission Requirements

Candidates must submit:

- GitHub repository
- README with complete instructions
- Brief design notes (how you structured backend, why)

---

6. Evaluation Criteria

Category	Weight
Code Quality & Structure	25%
Backend/API Design	25%
Google Ads Integration	20%
React UI/UX	20%
Documentation	10%