1. API Format Analysis & Calling Pattern Differences

A. URL and Authentication Methods

Bigin Systems (ColourCoats & Metalia):

Bigin

https://www.zohoapis.in/`/bigin/v1/{module_name}?page=1&per_page=200`

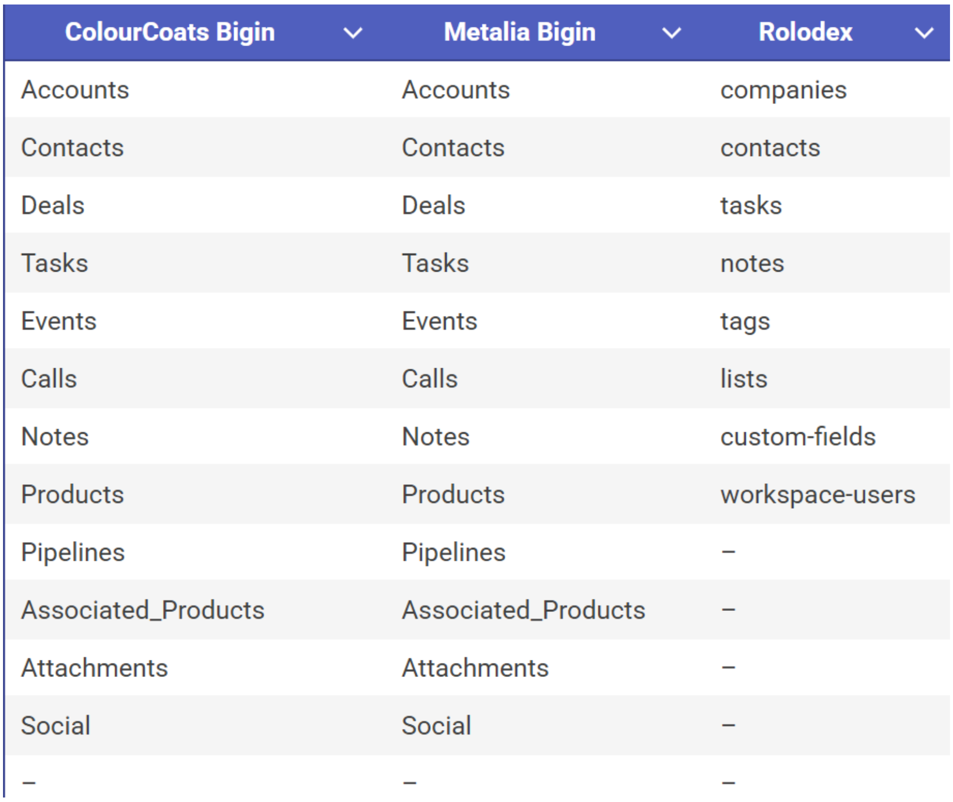
```
header

{

  "Authorization": "Zoho-oauthtoken your_token"  "Content-Type":
"application/json"

}
```

Rolodex

`https://api.rolodexcrm.com/user-api/v1/{module}?limit=100&offset=0`

```
headers

{

  "Content-Type": "application/json",

  "x-rolodex-api-key":
"bcdfb94a7e1c331bc201f0059011cfc31e0288af6b57434654f350986174694a"

}
```

1. Data Structure Analysis



ColourCoats Bigin	Metalia Bigin	Rolodex
Accounts	Accounts	companies
Contacts	Contacts	contacts
Deals	Deals	tasks
Tasks	Tasks	notes
Events	Events	tags
Calls	Calls	lists
Notes	Notes	custom-fields
Products	Products	workspace-users
Pipelines	Pipelines	–
Associated_Products	Associated_Products	–
Attachments	Attachments	–
Social	Social	–
–	–	–

Different table names in rolodexcrm and Bigin

B. ID Systems & Data Types

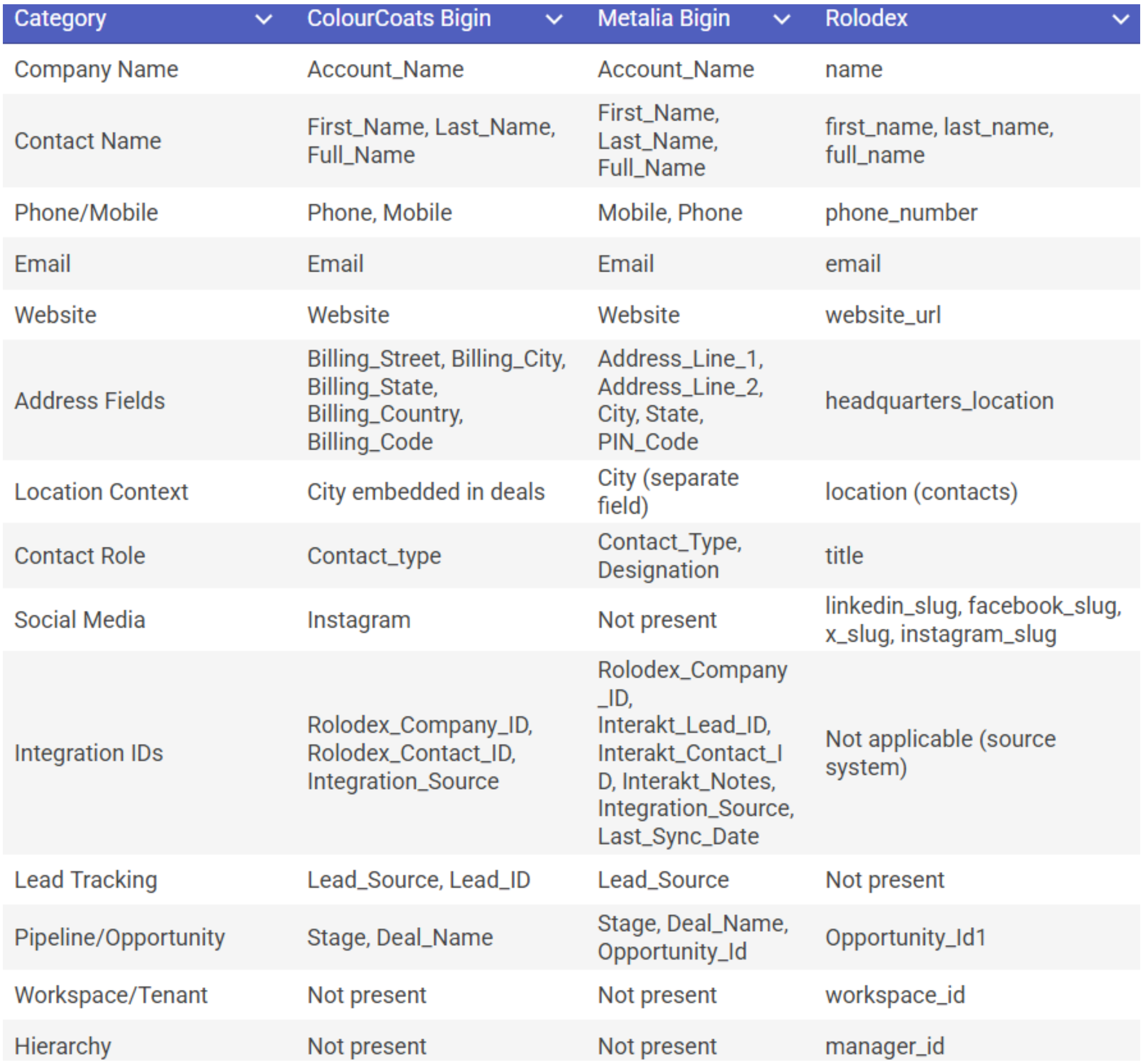
Bigin Systems (Both ColourCoats & Metalia):

- **Structure**: Heavily nested objects with relationship references

Rolodex:

- **Structure**: Flatter JSON with direct field access
- **Relationships**: Direct references or separate relationship endpoints

A. Field Naming Conventions



Category	ColourCoats Bigin	Metalia Bigin	Rolodex
Company Name	Account_Name	Account_Name	name
Contact Name	First_Name, Last_Name, Full_Name	First_Name, Last_Name, Full_Name	first_name, last_name, full_name
Phone/Mobile	Phone, Mobile	Mobile, Phone	phone_number
Email	Email	Email	email
Website	Website	Website	website_url
Address Fields	Billing_Street, Billing_City, Billing_State, Billing_Country, Billing_Code	Address_Line_1, Address_Line_2, City, State, PIN_Code	headquarters_location
Location Context	City embedded in deals	City (separate field)	location (contacts)
Contact Role	Contact_type	Contact_Type, Designation	title
Social Media	Instagram	Not present	linkedin_slug, facebook_slug, x_slug, instagram_slug
Integration IDs	Rolodex_Company_ID, Rolodex_Contact_ID, Integration_Source	Rolodex_Company_ID, Interakt_Lead_ID, Interakt_Contact_ID, Interakt_Notes, Integration_Source, Last_Sync_Date	Not applicable (source system)
Lead Tracking	Lead_Source, Lead_ID	Lead_Source	Not present
Pipeline/Opportunity	Stage, Deal_Name	Stage, Deal_Name, Opportunity_Id	Opportunity_Id1
Workspace/Tenant	Not present	Not present	workspace_id
Hierarchy	Not present	Not present	manager_id

Integration ID Fields Discovered:

ColourCoats Bigin:

- Rolodex_Company_ID: null (prepared for Rolodex company linking)
- Rolodex_Contact_ID: null (prepared for Rolodex contact linking)
- Integration_Source: null (tracks data origin system)

Metalia Bigin:

- Rolodex_Company_ID: null (prepared for Rolodex company linking)
- Interakt_Lead_ID: null (WhatsApp/chat platform integration)
- Interakt_Contact_ID: null (WhatsApp contact linking)
- Interakt_Notes: null (chat interaction history)
- Integration_Source: null (tracks data origin system)

Challenges issues

1. Null Values

- Across **Bigin (ColourCoats & Metalia)** and **Rolodex**, many fields are null or empty.
- Most fields in contacts and companies are either null or not meaningful for mapping.

2. Integration IDs Are Null

- In **Bigin (both ColourCoats and Metalia)**, the integration fields (`Rolodex_Company_ID`, `Rolodex_Contact_ID`) are mostly null.
- This prevents mapping between Bigin and Rolodex datasets.
- Without valid IDs, it is not possible to:
  - Identify common companies across datasets.
  - Identify employees working in multiple companies.
  - **Email & Phone Are Unreliable**
- Emails and phone numbers are not always consistent in one company. Cannot be used as a reliable key to link contacts.

3. Address Field Differences Between ColourCoats Bigin and Metalia Bigin

- **ColourCoats Bigin:**
  Accounts table contains detailed billing address fields:
      - `Billing_Street`, `Billing_City`, `Billing_State`, `Billing_Code`, `Billing_Country`.
  - These are mostly complete and usable as company addresses.
- **Metalia Bigin:**
  - Accounts table has different address fields:
    - `Address_Line_1`, `Address_Line_2` (mostly null), `City`, `State`, `PIN_Code`.
  - City and state are usable, but street-level address information is often missing.

Implication: Fields are inconsistent between the two Bigin systems, requiring normalization to a common format for mapping or analysis.

4. Integration IDs

- `Interakt_Part_ID` exists in **Metalia Bigin** but not consistently in **ColourCoats Bigin**.

5. Rolodex Location Field

- Rolodex only has a single `location` field.
- The value often combines **city, state, country** into one string.

- **Challenge:** Requires **normalization** to separate and standardize into usable fields (city, state, country) for mapping or comparison.

6. Name initials and inferred title

- Some contacts in ColourCoats Bigin and Metalia Bigin had **initials embedded in the name** that indicate roles (e.g., “Ar. shruti” → “Ar.” = Architect, “Id” = Interior Designer, “Cl” = Client).
  These initials were **not part of the actual name**, I implemented a logic to **filter out or interpret them**.

Solution Overview

I created 2 dedicated mapper files to standardize and unify heterogeneous data from three different CRM systems into consistent schemas. These mappers serve as the translation layer between disparate source formats and a unified data model.

Mapper Files Created

1. Unified_Companies_mapper.json

Maps company/account data from all three CRM sources to a standardized company schema.

2. unified_Personnel_mapper.json

Maps contact/personnel data from all three CRM sources to a standardized personnel schema.

For example

ColourCoats Bigin Address Handling

Maps structured billing address fields to unified schema:

- `Accounts.Billing_Street` → `address_street`
- `Accounts.Billing_City` → `address_city`
- `Accounts.Billing_State` → `address_state`
- `Accounts.Billing_Code` → `address_postal_code`
- `Accounts.Billing_Country` → `address_country`

Metalia Bigin Address Handling

Maps different address field structure to same unified fields:

- No street-level mapping (mostly null in source)
- `Accounts.City` → `address_city`
- `Accounts.State` → `address_state`
- `Accounts.PIN_Code` → `address_postal_code`

Rolodex Location Field Decomposition

Handles compound location strings through string splitting operations:

- `companies.headquarters_location.split(', ')[0]` → `headquarter_city`
- `companies.headquarters_location.split(', ')[1]` → `headquarter_state`

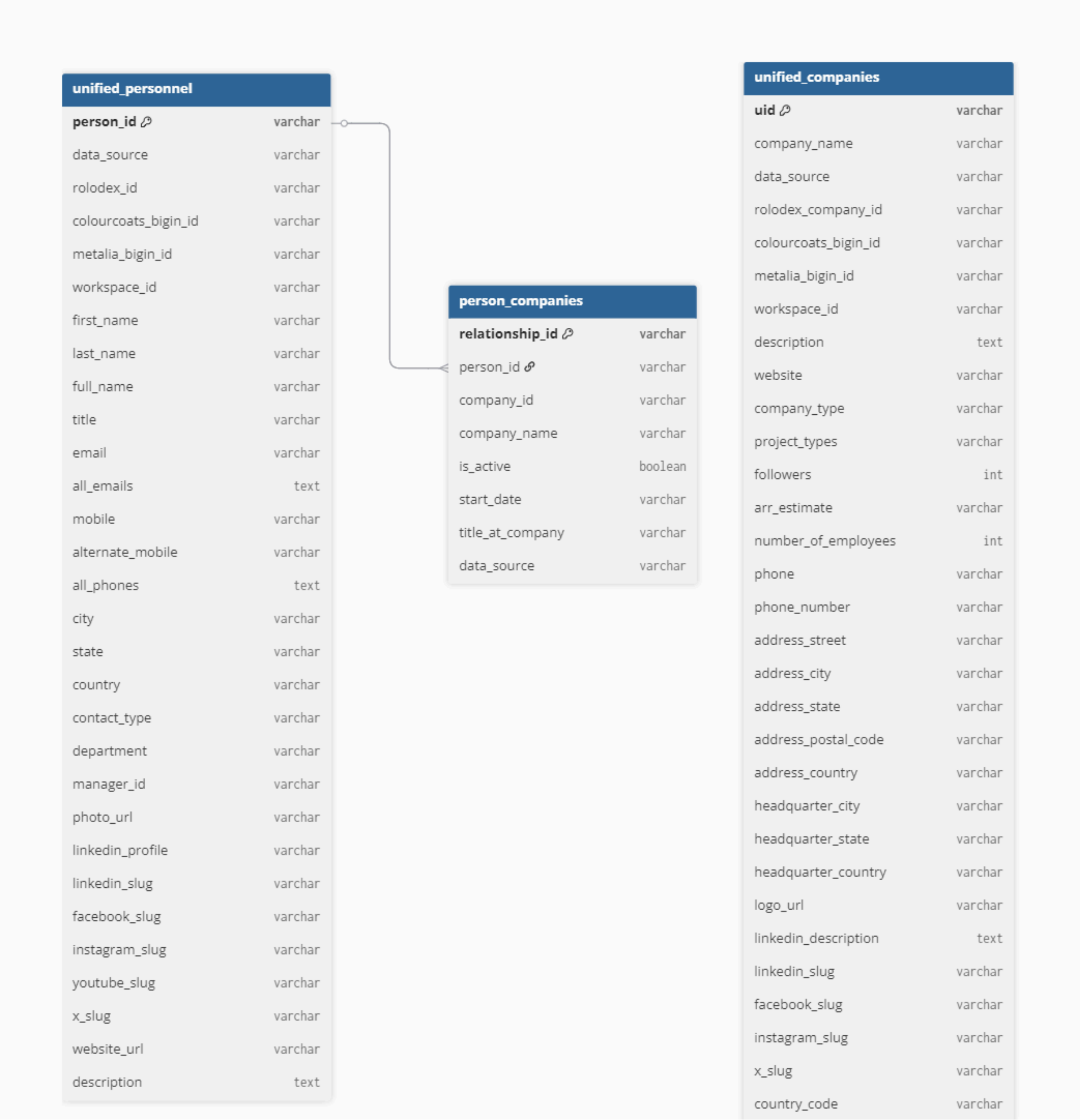
- `companies.headquarters_location.split(', ')[2]` → `headquarter_country`

Personnel Location Handling

Similar approach for personnel location data:

- Bigin: `Contacts.Mailing_City` → `city`
- Rolodex: `Rolodex_contacts.location.split(', ')[0]` → `city`
- Rolodex: `Rolodex_contacts.location.split(', ')[1]` → `state`
- Rolodex: `Rolodex_contacts.location.split(', ')[2]` → `country`

Unified schema/tables For firms and personnel



unified_personnel	
person_id	varchar
data_source	varchar
rolodex_id	varchar
colourcoats_bigin_id	varchar
metalia_bigin_id	varchar
workspace_id	varchar
first_name	varchar
last_name	varchar
full_name	varchar
title	varchar
email	varchar
all_emails	text
mobile	varchar
alternate_mobile	varchar
all_phones	text
city	varchar
state	varchar
country	varchar
contact_type	varchar
department	varchar
manager_id	varchar
photo_url	varchar
linkedin_profile	varchar
linkedin_slug	varchar
facebook_slug	varchar
instagram_slug	varchar
youtube_slug	varchar
x_slug	varchar
website_url	varchar
description	text

person_companies	
relationship_id	varchar
person_id	varchar
company_id	varchar
company_name	varchar
is_active	boolean
start_date	varchar
title_at_company	varchar
data_source	varchar

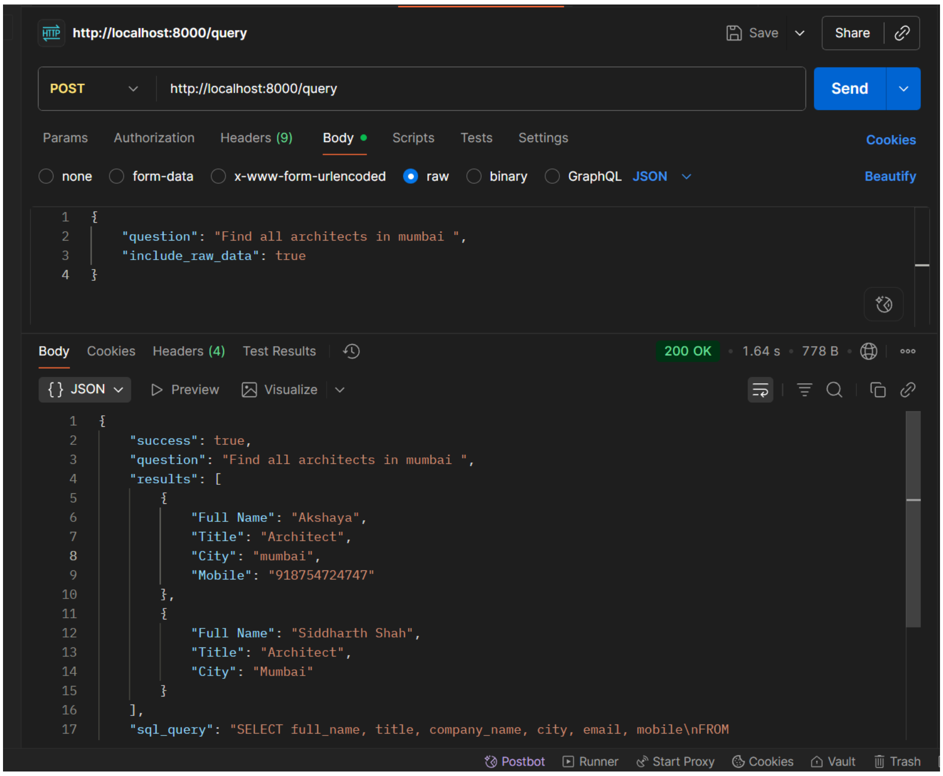
unified_companies	
uid	varchar
company_name	varchar
data_source	varchar
rolodex_company_id	varchar
colourcoats_bigin_id	varchar
metalia_bigin_id	varchar
workspace_id	varchar
description	text
website	varchar
company_type	varchar
project_types	varchar
followers	int
arr_estimate	varchar
number_of_employees	int
phone	varchar
phone_number	varchar
address_street	varchar
address_city	varchar
address_state	varchar
address_postal_code	varchar
address_country	varchar
headquarter_city	varchar
headquarter_state	varchar
headquarter_country	varchar
logo_url	varchar
linkedin_description	text
linkedin_slug	varchar
facebook_slug	varchar
instagram_slug	varchar
x_slug	varchar
country_code	varchar

Task 3.

Implemented and integrated a FastAPI backend with natural language processing (NLP) capabilities, enabling efficient query handling and scalable API services.

The system converts plain English queries into SQL queries using **Google Gemini AI** and returns structured JSON responses. This serves as the backend API that could be integrated with WhatsApp or any chat interface.

Backend: http://localhost:8000/query
Front end : http://localhost:5173/



POST http://localhost:8000/query

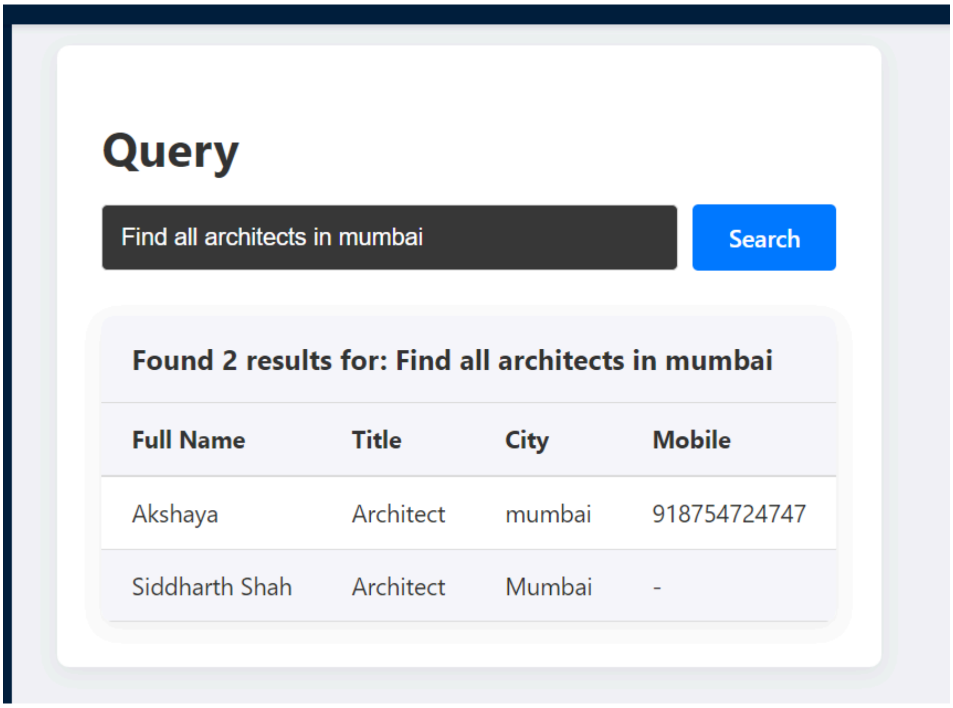
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "question": "Find all architects in mumbai ",
    "include_raw_data": true
}
```

{} JSON

```
{
    "success": true,
    "question": "Find all architects in mumbai ",
    "results": [
        {
            "Full Name": "Akshaya",
            "Title": "Architect",
            "City": "mumbai",
            "Mobile": "918754724747"
        },
        {
            "Full Name": "Siddharth Shah",
            "Title": "Architect",
            "City": "Mumbai"
        }
    ],
    "sql_query": "SELECT full_name, title, company_name, city, email, mobile\nFROM
```

Postman API endpoint



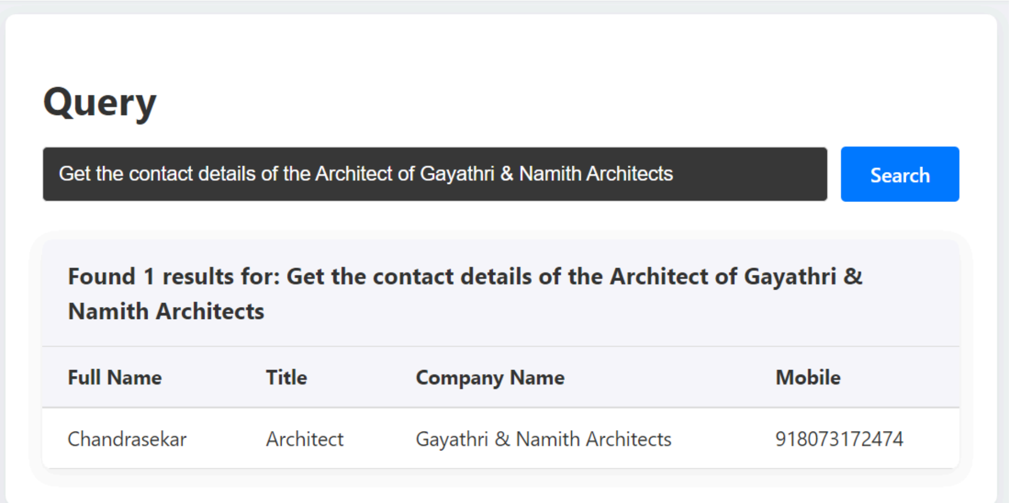
Query

Find all architects in mumbai

Found 2 results for: Find all architects in mumbai

Full Name	Title	City	Mobile
Akshaya	Architect	mumbai	918754724747
Siddharth Shah	Architect	Mumbai	-

Fig1 Frontend UI 1



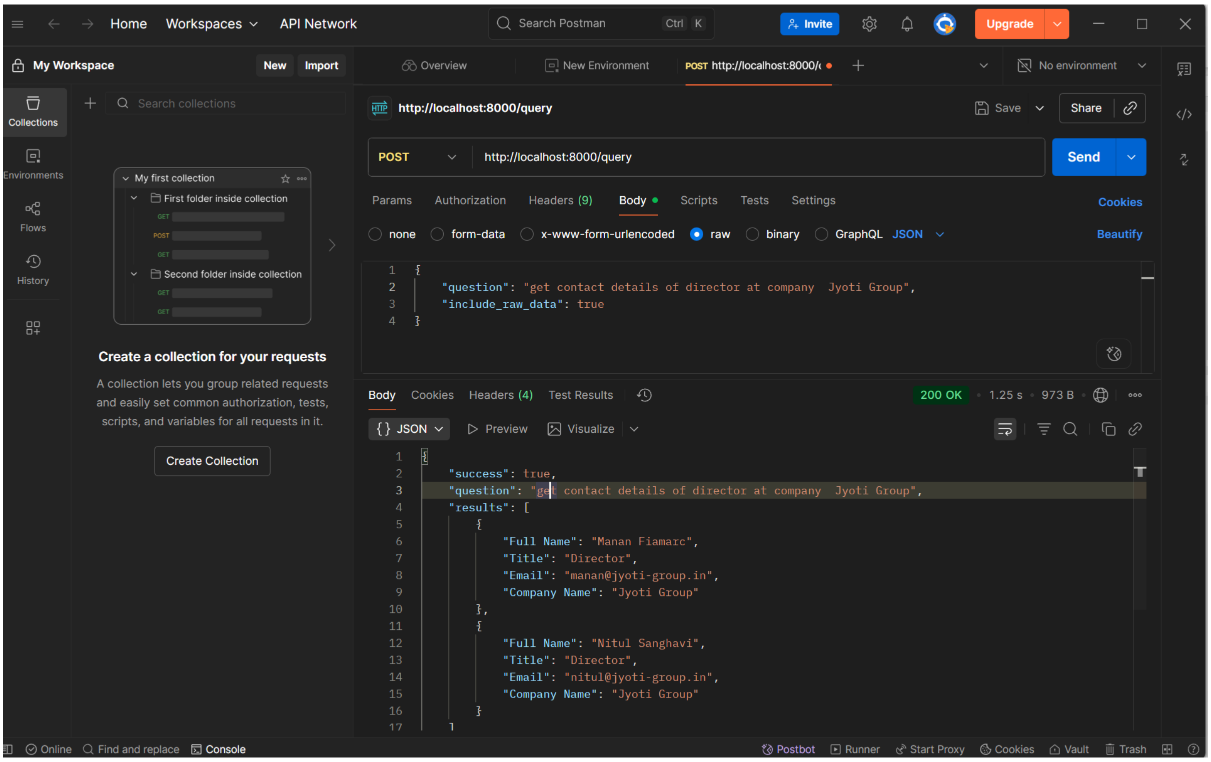
Query

Get the contact details of the Architect of Gayathri & Namith Architects

Found 1 results for: Get the contact details of the Architect of Gayathri & Namith Architects

Full Name	Title	Company Name	Mobile
Chandrasekar	Architect	Gayathri & Namith Architects	918073172474

Fig Frontend UI



POST http://localhost:8000/query

```
{
    "question": "get contact details of director at company  Jyoti Group",
    "include_raw_data": true
}
```

200 OK • 1.25 s • 973 B

```
{
    "success": true,
    "question": "get contact details of director at company  Jyoti Group",
    "results": [
        {
            "Full Name": "Manan Fiamarc",
            "Title": "Director",
            "Email": "manan@jyoti-group.in",
            "Company Name": "Jyoti Group"
        },
        {
            "Full Name": "Nitul Sanghavi",
            "Title": "Director",
            "Email": "nitul@jyoti-group.in",
            "Company Name": "Jyoti Group"
        }
    ]
```