**Motorq Backend Assignment Round**

**Name-**Ishita Agarwal

**Creating table in Airtable**

A screenshot of a computer

Description automatically generated

**Generating requests from CoinGecko website.**

A screenshot of a computer

Description automatically generated

**Background job functions**

Setting Up API Endpoints and Import Statements:

I have defined the API endpoints for CoinGecko's /coins/list and /simple/price endpoints and also imported necessary modules: requests for making HTTP requests, schedule for scheduling tasks, and time for adding a delay.

Defining Base ID and Table Name:

I have set the BASE\_ID to the Airtable base which I want to work with, and TABLE\_NAME to the specific table within that base.

Update Coin Details Function (update\_coin\_details):

This function is intended to update coin details in your Airtable database for a list of coins provided. It iterates through the list of coins and uses the pyairtable library to update the corresponding records in the Airtable table.

Update Current Prices Function (update\_current\_prices):

This function is intended to update the current coin prices in your Airtable database. It iterates through the provided prices dictionary (assuming it's in the format returned by the CoinGecko API). For each coin ID, it constructs a record update with the current price in USD and uses the pyairtable library (which is missing in the code) to update the corresponding records in the Airtable table.

Background Job Functions:

fetch\_and\_update\_coin\_details: This function fetches the list of coins from CoinGecko's /coins/list API. If successful, it extracts the top 20 coins and calls the update\_coin\_details function to update the Airtable records with their details.

fetch\_and\_update\_current\_prices: This function fetches current prices for the specified coin IDs from CoinGecko's /simple/price API. If successful, it calls the update\_current\_prices function to update the Airtable records with the current prices.

Scheduling Background Jobs:

The schedule library is used to schedule the execution of the background job functions. The .do(...) method specifies the function to be executed at a specific interval (10 minutes for coin details and 1 minute for current prices).

Running Scheduled Tasks:

The script enters an infinite loop where it checks if there are any scheduled tasks to run using schedule.run\_pending(). It then adds a small delay using time.sleep(1) to avoid high CPU usage while waiting for the next iteration.

**CODE:-**

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

**OUTPUT:-**

A screenshot of a computer program

Description automatically generated

**Explanation:-**

**Imports:**

requests: This library is used to make HTTP requests to the CoinGecko API.

json: This library is used to parse JSON data.

os: This module provides a way to use operating system-dependent functionality, here it's used to execute the notify-send command.

time: This module provides various time-related functions.

Infinite Loop (while True):

The code is enclosed in an infinite loop, which means it will keep executing the block of code within the loop indefinitely.

**API Request:**

The script makes an HTTP GET request to the CoinGecko API to fetch the current price of Bitcoin in USD. The API endpoint used is https://api.coingecko.com/api/v3/simple/price?ids=bitcoin&vs\_currencies=usd.

requests.get(...).text fetches the response content from the API as text.

JSON Parsing:

json.loads(t) parses the JSON response obtained from the API request.

The parsed JSON data is stored in the t variable.

**Extracting Price:**

The code extracts the Bitcoin price in USD from the parsed JSON data using t["bitcoin"]["usd"].

The extracted price is converted to an integer using int(...), assuming that the API returns the price as a string.

**Desktop Notification:**

The os.system(...) line uses the notify-send command to display a desktop notification with the Bitcoin price. The -t 10000 flag sets the notification timeout to 10 seconds (10,000 milliseconds).

The f"Price is {price} " part of the command includes the fetched Bitcoin price in the notification message.

Sleep:

After displaying the notification, the script sleeps for 60 seconds using time.sleep(60). This introduces a delay between each iteration of the loop. This is likely intended to prevent excessive requests to the CoinGecko API and notification spam.

The loop continues indefinitely, fetching the Bitcoin price every 60 seconds and displaying a desktop notification with the updated price.

The code (below) continuously fetches the current price of Bitcoin in USD from the CoinGecko API, displays a desktop notification with that price using the notify-send command (assumed to be a Linux command), and then waits for 60 seconds before repeating the process.

CODE:-

A computer screen shot of text

Description automatically generated

Implementing a simple rate limiting mechanism to ensure that requests to the CoinGecko API are within the allowed limits.

A computer screen shot of a program code

Description automatically generated

**Exposed APIs:**

* Flask app with two endpoints: /coins and /coins/price/:coinId
* Retrieves cryptocurrency data from an Airtable database
* Returns cryptocurrency information as JSON at /coins endpoint
* Fetches current coin price from Redis cache at /coins/price/:coinId
* If cached, returns the price; if not, queries Airtable and caches the price
* Airtable fields used: "Name," "Symbol," "MarketCap," and "Current Price"
* Utilizes Redis for caching to minimize API calls
* The cache has a time-to-live (TTL) of 10 minutes (600 seconds)
* Replace placeholders like BASE\_ID, REDIS\_HOST, and field names
* Requires pyairtable and redis libraries to be installed

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

OUTPUT:-

A screen shot of a computer

Description automatically generated