

Zepto

<u>Aa</u> Title	<u>☰</u> Tags
<u>Medium</u>	App
<u>Type of scraper</u>	HTTP Programming
<u>Type of crawling</u>	Generic
<u>Depth and Scraping Strategy</u>	BFS Nested
<u>Difficulty</u>	Intermediate
<u>Technology</u>	GenyMotion HTTP Toolkit Python

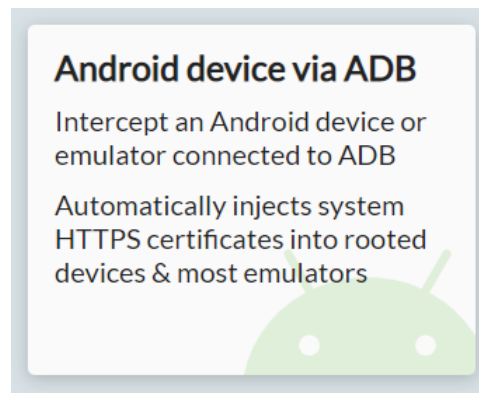
Zepto is a Mumbai-based startup that operates a 10-minute instant grocery delivery service. The company holds a massive database of daily groceries. Our target is to acquire this data.

We will achieve data acquisition by scraping the Zepto app available on the google play store. The following is a detailed process of how we acquire the raw material to implement the task.

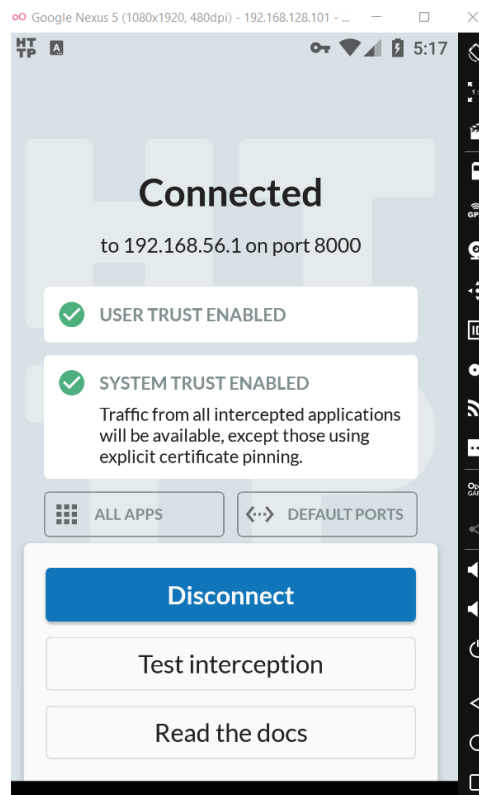
Process

1. We will use GenyMotion for a rooted android device, HTTP Toolkit for runtime injection, and Python to scrape data.
2. Download the Zepto apk from [here](#).
3. Open the android emulator in GenyMotion.
4. Drag and drop the Zepto apk file in the android emulator to install the apk.























5. Open the HTTP Toolkit. Choose the option “Android Device via ADB”. It will automatically connect to the emulator. You can verify it on the emulator.



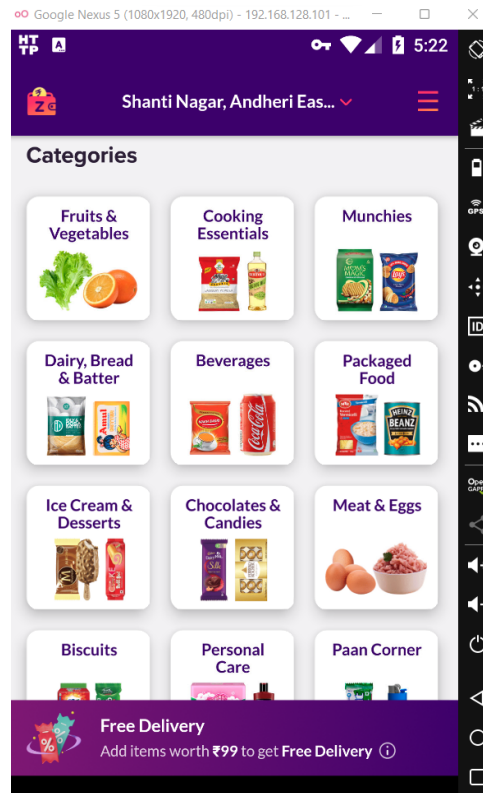
6. To verify whether our emulator device is successfully connected to the HTTP Toolkit, we can confirm the connection on the emulator itself.



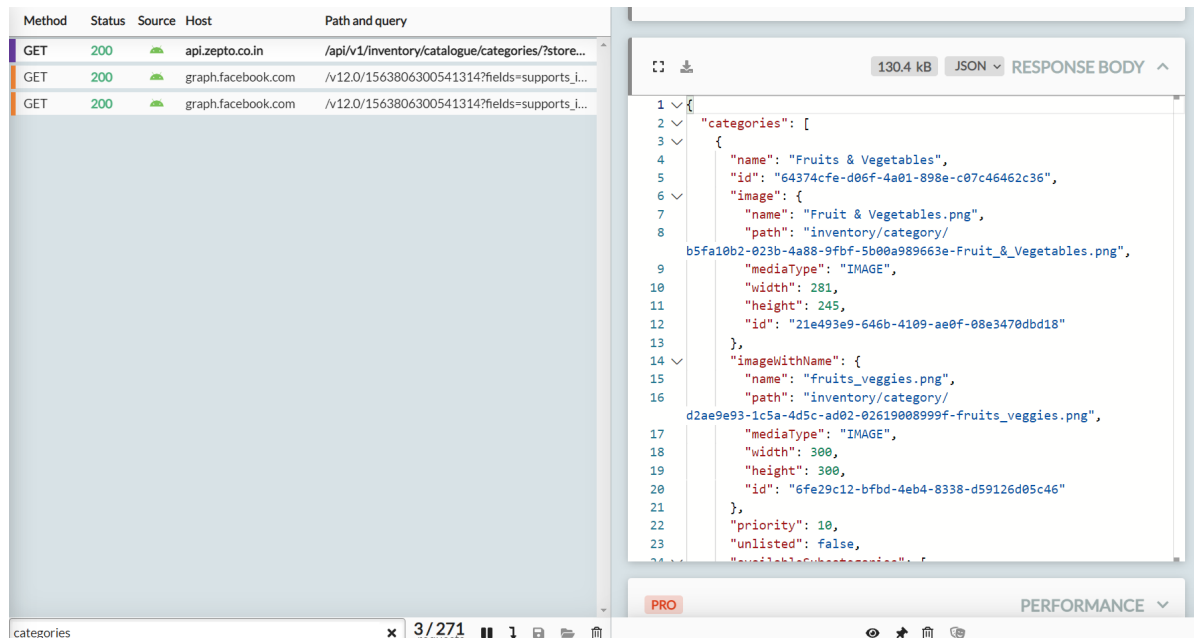
7. Open the Zepto app on the emulator.
8. As soon as we start the app, we can see the requests being recorded on the HTTP Toolkit.

Method	Status	Source	Host	Path and query
GET	200		api.zepto.co.in	/api/v1/user/customer/check_update/?app_ver...
POST	200		eu1.clevertap-prod.com	/a1?os=Android&t=40400&z=65W-5R5-4R6Z...
POST	200		sentry.zepto.co.in	/api/9/envelope/
GET	200		api.zepto.co.in	/api/v1/user/config/version/?current_version=1
POST	200		sentry.zepto.co.in	/api/9/envelope/
GET	200		api.mixpanel.com	/decide?version=1&lib=android&token=dcc87...
POST	200		api.zepto.co.in	/api/v2/user/customer/info/
GET	200		codepush.appcenter.ms	/v0.1/public/codepush/update_check?deploy...
GET	200		clientservices.googlea...	/chrome-variations/seed?osname=android_we...
GET	200		api.zepto.co.in	/api/v2/user/config/values/
GET	200		api.zepto.co.in	/api/v1/user/config/promotions/
GET	200		api.zepto.co.in	/api/v1/user/customer/addresses/
POST	200		onelink.appsflyer.com	/shortlink-sdk/v2/tC90
GET	200		api.zepto.co.in	/api/v1/user/customer/addresses/
GET	200		codepushupdates.azu...	/storagev2/7tuzSWFCXHbFj9q-_FINMdJ3gRb...
POST	200		launches.appsflyer.com	/api/v6.3/androidevent?app_id=com.zeptocons...
POST	200		inapps.appsflyer.com	/api/v6.3/androidevent?app_id=com.zeptocons...
GET	304		jp-remote-assets.s3.a...	/juspay/hyper-os/in.juspay.hyperos/release/2....
POST	200		eu1.clevertap-prod.com	/a1?os=Android&t=40400&z=65W-5R5-4R6Z...
POST	200		eu1.clevertap-prod.com	/a1?os=Android&t=40400&z=65W-5R5-4R6Z...
GET	200		assets.juspay.in	/juspay/payments/2.0/release/v1-config.zip
GET	200		assets.juspay.in	/juspay/payments/release/sdk_config.json

9. We can search for the Zepto APIs in the search bar in the bottom left corner.
10. Switch to the emulator. You will find a section called “Categories” while you scroll down. We can guess that most of the data would have been stored across various categories.



11. Thus, search for “categories” in HTTP Toolkit. And the result shows exactly 1 API that extends all the necessary data for the categories.

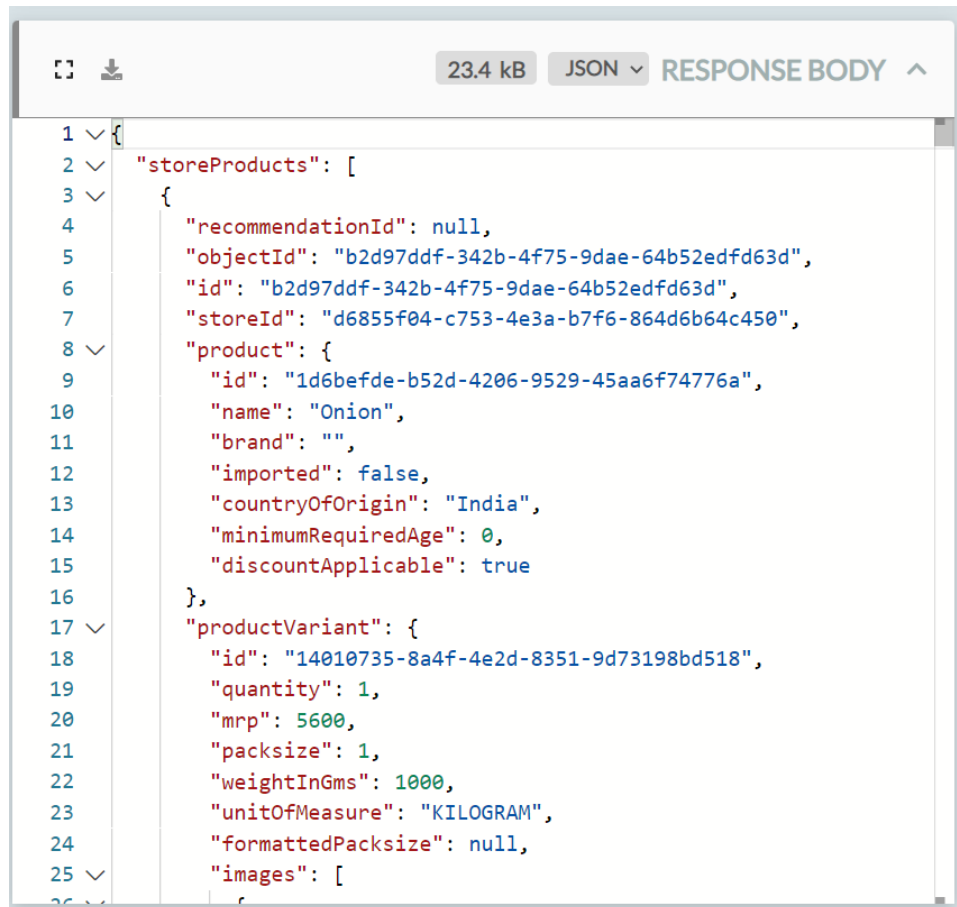


12. In step 11, we can conclude from the response body that it matches the data from step 10.
13. We need to extract 2 important artifacts from the former steps, viz., request URL, and Headers.



14. After exploring categories, our task is to further explore and loop through each category. This will enable us to scrap products. Switch back to the emulator. Navigate to the first category, i.e., "Fruits and Vegetables".

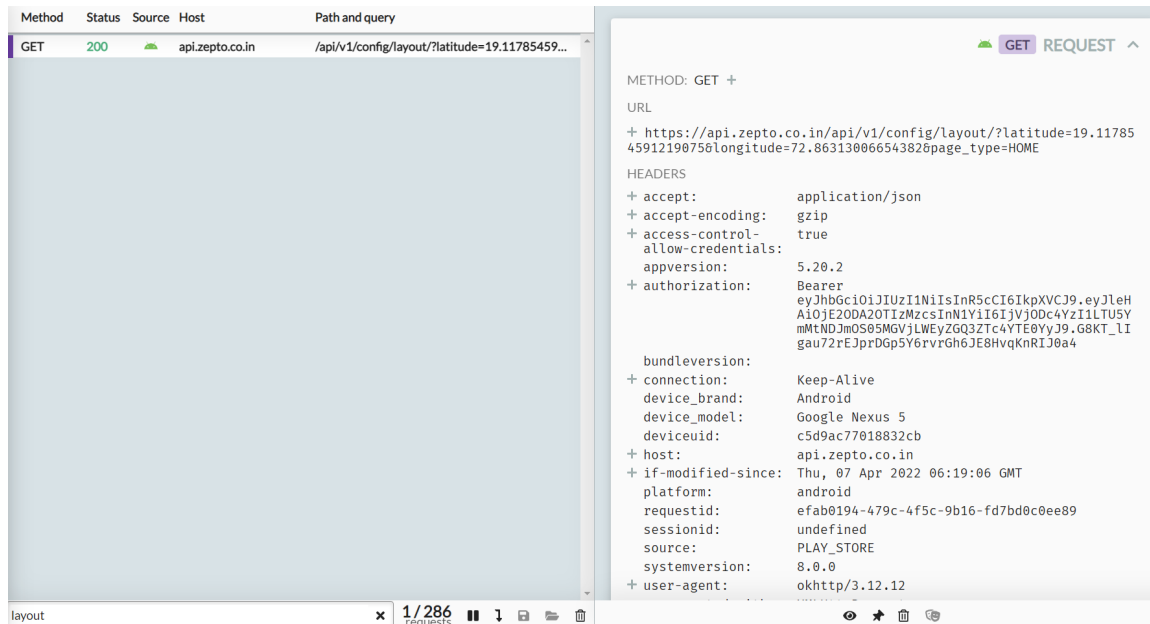
16. Here, after exploring a few APIs, the last API gives us the expected result. We'll copy the request URL.



```
1 {
2   "storeProducts": [
3     {
4       "recommendationId": null,
5       "objectId": "b2d97ddf-342b-4f75-9dae-64b52edfd63d",
6       "id": "b2d97ddf-342b-4f75-9dae-64b52edfd63d",
7       "storeId": "d6855f04-c753-4e3a-b7f6-864d6b64c450",
8       "product": {
9         "id": "1d6befde-b52d-4206-9529-45aa6f74776a",
10        "name": "Onion",
11        "brand": "",
12        "imported": false,
13        "countryOfOrigin": "India",
14        "minimumRequiredAge": 0,
15        "discountApplicable": true
16      },
17      "productVariant": {
18        "id": "14010735-8a4f-4e2d-8351-9d73198bd518",
19        "quantity": 1,
20        "mrp": 5600,
21        "packsize": 1,
22        "weightInGms": 1000,
23        "unitOfMeasure": "KILOGRAM",
24        "formattedPacksize": null,
25        "images": [
26          ...

```

17. We can match “Onion” as the first product from steps 14 and 16. Thus, the API spotted is correct. Our last task is to find the location-wise search. Most APIs use “layout” to name location-based search APIs. We'll search for the same.



18. There is just a single API URL that gives us the result about the location by calculating latitudes and longitudes. We shall copy the URL.

Here we commence are raw material extraction process. We have:

- categories API URL
- products API URL
- location configuration API URL

Before moving to the implementation, we shall understand the characteristics of our scraping approach for Zepto.

- As we are interacting with API URLs using HTTP requests, the data scraper is classified into HTTP Programming based scraper.
- Our objective is to maximize the data scraped. So, we have set no filters on the crawler. Thus, we classify it as a generic crawling.

- First, we scraped categories and later products were derived from categories (we'll clarify in the implementation process). This makes the scraper explore 2 different layers in BFS fashion. Thus, the scraper is nested and follows the BFS fashion to scrape data.

Implementation

1. For this implementation, we shall write a python script to make HTTP requests to GET data from the Zepto database. We'll initiate by assigning the category, products, and location URLs to variables.

```
locationApiUrl = "https://api.zepto.co.in/api/v1/config/layout/"
subApiUrl = "https://api.zepto.co.in/api/v2/inventory/catalogue/store-products/"
storeApiUrl = "https://api.zepto.co.in/api/v1/inventory/catalogue/categories/"
```

2. For the location, we understand that by adding values of latitude and longitude to locationApiUrl in the fashion shown in Process(step 18), we can configure our location.

```
locationApiUrl + "?" + "latitude=" + latitude + "&" + "longitude=" + longitude + "&" + "page_type=HOME"
```

3. On request from step 2, we receive Store ID in response which shall be used to extract categories in that location.

```
storeApiUrl + "?" + "store_id=" + storeID
```

4. After extracting the categories, we can make a request to extract products.

```
subApiUrl + "?" + "store_id=" + storeID + "&" + "subcategory_id=" + subcategoryID + "&" + "page_number=" + str(pageNumber)
```

To iterate through multiple pages, we have used `page_number = str(pageNumber)`. This would allow us to increment `pageNumber` and go to the next page.

5. After mapping the props and data points from APIs to python variables, we can store all the data in lists which shall then be dumped into an excel sheet using

Pandas.

```
Done scraping : Fruits & Vegetables
Done scraping : Cooking Essentials
Done scraping : Munchies
Done scraping : Dairy, Bread & Batter
Done scraping : Beverages
Done scraping : Packaged Food
Done scraping : Ice Cream & Desserts
Done scraping : Chocolates & Candies
Done scraping : Meats, Fish & Eggs
Done scraping : Biscuits
Done scraping : Personal Care
Done scraping : Paan Corner
Done scraping : Home & Cleaning
Done scraping : Health & Hygiene
Done scraping : Curated For You
```

The following is the snapshot of the extracted data:

	name	mrp	discountPercent	availableQuantity	discountedSellingPrice	weightInGms	outOfStock	quantity
0	Onion	2300	17	3	1900	1000	False	1
1	Tomato Hybrid	4000	15	3	3400	1000	False	1
2	Tender Coconut	5100	15	3	4300	58	False	1
3	Ladies Finger	1400	14	3	1200	250	False	250
4	Coriander Leaves	2000	15	3	1700	100	False	100
5	Potato	3200	15	3	2700	1000	False	1
6	Lemon	7500	16	3	6300	200	False	200
7	Watermelon	5400	16	3	4500	58	False	1
8	Banana Robusta	2900	17	3	2400	348	False	6
9	Chilli Green	1700	17	3	1400	100	False	100

The entire dataset can be accessed [here](#).

Thank you!

