Plaid API

Overview

- API access

  To gain access to the Plaid API, create an account on the Plaid Dashboard. Once you have completed the signup process and acknowledged our terms, you will be provided with a live `client_id` and `secret` via the Dashboard.

- API headers

  Almost all Plaid API endpoints require a `client_id` and `secret`. The APIs implemented and deployed in this doc all require headers to be configured with `PLAID-CLIENT-ID` and `PLAID-SECRET`.

- API host

  Plaid has three environments: Sandbox, Development, and Production. In this implementation, we will be using **sandbox**, which provides test credentials and life-like data. For detailed differences, visit this official documentation.

```
https://sandbox.plaid.com (Sandbox)
https://development.plaid.com (Development)
https://production.plaid.com (Production)
```

The Plaid Flow

For a user to interact with their banks securely via Plaid, there are several steps:

- **Step 1**: The client requests a Link Token from the application
  - a Link token is a user interface component, which is used as a secure way to initiate the Plaid Link process
- **Step 2**: The application requests (along with my API's secret) a Link Token from Plaid server, and return the Link Token back to client
- **Step 3a**: The client talks directly to Plaid with the Link Token, and Plaid communicates with client's bank
- **Step 3b**: Plaid will generate an access token/password to the client for each bank requested, so the client can retrieve information in the future without needing to log in again
- **Step 4**: Plaid sends back the token, called public (or once used, dirty) token, to the client
- **Step 5**: Client sends the public token to the application
- **Step 6a**: The application sends public token back to Plaid (along with secret)
- **Step 6b**: After validating the public token has not been used before, Plaid will create an access token; Plaid sends the access token to the application, which will be stored in a database

Refer to this official quick start tutorial for detailed information.

APIs related to Link Token (Step 1 - Step 3a) are not required nor implemented. Step 3b is implemented in the `createPublicTokens` . Step 4 - Step 6 are simplified and implemented in the `exchangePublicTokensAndStore` API.

API Reference

- **createPublicTokens**
  - Creates new public tokens for a set of predefined institution IDs and clears any existing tokens from the Firestore. In the deployed version, Chase(ins_56), Bank of America (ins_127989), Capital One (ins_128026) are used
  - POST - https://us-central1-budgetflow-yosun.cloudfunctions.net/createPublicTokens
  - Body implicitly used by Plaid's provided endpoint (*won't be needed in our POST request*)

```
{
    "client_id": "YOUR_PLAID_CLIENT_ID",
    "secret": "YOUR_PLAID_SECRET",
    "institution_id": "ins_56", // Chase's institution_id
    "initial_products": ["transactions"]
}
```

- **exchangePublicTokensAndStore**
  - Exchanges public tokens for access tokens and updates the Firestore `accessTokens` collection.
  - POST - https://us-central1-budgetflow-yosun.cloudfunctions.net/exchangePublicTokensAndStore
  - Body implicitly used by Plaid's provided endpoint (*won't be needed in our POST request*)

```
{
    "client_id": "YOUR_PLAID_CLIENT_ID",
    "secret": "YOUR_PLAID_SECRET",
    "public_token": "YOUR_PUBLIC_TOKEN"
}
```

- **getTransactions**
  - Fetches transactions within a specified date range for each access token stored in Firestore, then stores these transactions in the Firestore `transactions` collection.
  - POST - https://us-central1-budgetflow-yosun.cloudfunctions.net/getTransactions
  - Body implicitly used by Plaid's provided endpoint (*won't be needed in our POST request*)

```
{
    "client_id": "YOUR_PLAID_CLIENT_ID",
    "secret": "YOUR_PLAID_SECRET",
    "access_token": "YOUR_ACCESS_TOKEN",
    "start_date": "2024-01-01",
    "end_date": "2024-02-29"
}
```

- **calculateMonthlyBudget**
  - Calculates the total expenditure per category from transactions stored in Firestore and estimates monthly budgets by dividing the totals by a predefined number.
  - POST - https://us-central1-budgetflow-yosun.cloudfunctions.net/calculateMonthlyBudget
  - Body

```
{
    "start_date": "2024-01-01",
    "end_date": "2024-02-29"
}
```

- **InstitutionsSearchRequest**

- Provided, not Implemented.
- Returns a JSON response containing details for institutions that match the query parameters, up to a maximum of ten institutions per query.
- See this for official documentation.
- POST - https://sandbox.plaid.com/institutions/search

```
{
  query: institutionID,
  products: ['transactions'],
  country_codes: ['US'],
}
```

Usage

If you would like to implement the APIs, follow these steps for setup:

- Log in from your terminal `$ firebase login`; this will redirect you to google's login page
- Install firebase tool: `$ install -g firebase-tools`
- Initialize your firebase project: `$ firebase init`; choose Firestore and Functions for database and API deployment
- Install dependencies:
  `$ cd functions`
  `$ npm install plaid@latest`
- Modify index.ts, then deploy the functions to firebase: `$ firebase deploy`

You can start testing the APIs using Postman; note that the requests' body aforementioned are wrapped in our APIs; only `getTransactions` will require start_date and end_date

- **createPublicTokens**
  - POST - https://us-central1-"YOUR_PROJECT_NAME".cloudfunctions.net/createPublicTokens
  - At this point, you should see collection "publicTokens" created, and documents generated in firestore
- **exchangePublicTokensAndStore**
  - POST - https://us-central1-"YOUR_PROJECT_NAME".cloudfunctions.net/exchangePublicTokensAndStore
  - At this point, you should see collection "accessTokens" created, and documents generated in firestore
- **getTransactions**
  - POST - https://us-central1-"YOUR_PROJECT_NAME".cloudfunctions.net/getTransactions
  - Body (modify as needed)

```
{
    "start_date": "2024-01-01",
    "end_date": "2024-02-29"
}
```

  - At this point, you should see collection "accessTokens" created, and documents generated in firestore
- **calculateMonthlyBudget**
  - POST - https://us-central1-"YOUR_PROJECT_NAME".cloudfunctions.net/calculateMonthlyBudget
  - Sample response:

```
{
    "success": true,
    "monthlyBudgets": {
        "Food and Drink": 3317.189999999999,
        "Travel": 35.19,
        "Payment": 6310.5,
```

```
        "Transfer": 20537.34,
        "Recreation": 235.5,
        "Shops": 1500
    }
}
```