Business Manger API Project

**Summary**: Develop a Node.js script to track daily spending across 50,000+ Facebook ad accounts. Utilize MySQL for data storage and Google Sheets for visualization. Maintain data accuracy by handling confirmed and unconfirmed spending. Manage tokens, Business Managers, and account changes dynamically.

**Folder Structure:** Our project is organized into a clear and structured directory layout to ensure efficient development and maintenance. The structure is designed to enhance modularity, facilitate collaboration, and ensure easy access to different components of the project.

project-root/

├─ module/

│ ├─ businessmanagers.js

│ ├─ config.js

│ ├─ tabtosheet.js

│ ├─ tempFinal.js

├─ src/

│ ├─ JSON/

│ │ ├─ config.json

│ ├─ index.js

├─ package.json

└─ package-lock.json

1. **`module` Directory:** This directory houses essential modules that form the core functionality of our application. Each module serves a specific purpose, allowing us to compartmentalize and manage our code more effectively.
   1. businessmanagers.js: Handles the logic related to managing business managers Id and their data. It copies all data from BM google sheet to our database table business\_mangers Table.
   2. config.js: Manages configuration settings for the project, including database and Google Sheet configurations.
   3. tabtosheet.js: It handles the google sheet to get updated data from our database from spendinfo\_temp and spendinfo\_final tables. It just copy those two table and past it to the google sheet.
   4. tempFinal.js: Manages the temporary and final data storage and processing to our database tables. It stores all data from facebook BM to our spendinfo\_temp and spendinfo\_final tables..
2. **`src` Directory:** The src directory contains our main source code files. It's the heart of our application and where the main execution begins.
   1. JSON Directory: A subdirectory within src that houses configuration files
   2. Notably, config.json is stored here to configure our interaction with Google Sheets.
3. **package.json and package-lock.json:** These files define our project's dependencies, scripts, and metadata. They ensure that we are using consistent versions of libraries and packages across development environments.
4. **Index.js:** This is our primary entry point for the application. It orchestrates the execution of various modules and handles scheduling tasks

**Benefits of Our Project Structure:**

1. **Modularity and Reusability:** By separating functionality into distinct modules, we achieve code modularity and reusability, making it easier to maintain and extend the application over time.
2. **Organization and Clarity:** Our structured layout provides a clear separation of concerns, making it simpler to understand and manage the project components.
3. **Efficient Collaboration:** The clear directory structure promotes collaborative development. Team members can work independently on different modules without interfering with each other's work.
4. **Ease of Maintenance:** This organization ensures that troubleshooting, debugging, and updates are efficient. Changes in one module are less likely to impact others.
5. **Enhanced Scalability:** With our modular approach, adding new features or expanding functionality becomes more straightforward. New components can be integrated without causing disruption.

By adopting this well-organized project structure, we aim to deliver a reliable, maintainable, and scalable solution that meets your business needs effectively.

**Database Structure Overview:** Our project's backend is powered by a well-structured MySQL database that efficiently manages data related to Facebook ad accounts, spending information, errors, and configurations. The database schema is designed to ensure data integrity, optimal querying, and seamless interaction with our application components.

1. **`adsaccount` Table:** This table stores information about individual ad accounts and their associated details, including account IDs, Business Manager IDs, account names, and access tokens. This table data won’t be deleted for no time. Since sometime facebook remove the ads account from BM that’s why we store those ad account in this table as per as our project requirement.
2. **business\_managers Table:** The business\_managers table maintains records of various Business Managers and their respective data. This includes Business Manager names, associated account IDs, and access tokens. This table is a copy of our BM google sheet that we have.
3. **error\_logs Table:** This table records error messages and associated timestamps, helping us keep track of any issues that occur during data processing or interaction with external services.
4. **spendinfo\_final Table:** The spendinfo\_final table stores final spending data collected from Facebook Ads API. It includes attributes such as account names, account IDs, Business Manager names, spending details, dates, and additional contextual information. The data of this table won’t be deleted for no time. All data that contain in this table are 3 days ago from current time and save to this final table.
5. **spendinfo\_temp Table:** Similar to spendinfo\_final, the spendinfo\_temp table temporarily holds spending data during processing before it's finalized. This table contain all data that are within last 3 days and if it overs 3 days then it will move to spendinfo\_final table. This table continuously **update every single hour**.

**Key Advantages of Our Database Structure:**

1. **Data Segmentation:** Our tables are organized to store different types of data separately, promoting a clean and structured database schema.
2. **Efficient Queries:** With appropriate indexing and relational design, our database supports efficient querying and retrieval of relevant data.
3. **Data Integrity:** Primary keys and indexes ensure data integrity and uniqueness within the tables, preventing duplicate or inconsistent data.
4. **Modularity:** The separation of tables into distinct categories allows us to manage specific data types without compromising the overall database structure.
5. **Scalability:** Our well-designed database structure is built to scale, allowing us to accommodate a growing volume of data and user interactions without sacrificing performance.
6. **Error Tracking:** The error\_logs table ensures that any issues encountered during data processing are recorded and can be reviewed for troubleshooting.

By adhering to this organized and optimized database structure, we ensure that our application can effectively manage, process, and retrieve data to provide you with accurate and valuable insights.

**Libraries used in this:** You have used the following libraries and packages for your project:

1. **googleapis (^126.0.0):**

This library allows you to interact with various Google services and APIs, including Google Sheets which you are using for visualization.

1. **moment (^2.29.4):**

A popular library for working with dates and times in JavaScript. It provides utilities for parsing, formatting, and manipulating dates and times.

1. **mysql (^2.18.1):**

The MySQL library for Node.js that enables you to interact with MySQL databases, allowing you to store and retrieve data.

1. **node-cron (^3.0.2):**

A scheduling library that helps you run scheduled tasks at specified intervals. You are using this to run your script at regular intervals for data retrieval and updates.

1. **node-fetch (^3.3.2):**

A lightweight library that enables you to make HTTP requests in Node.js, which you are using to fetch data from external APIs (like Facebook Ads API).

These libraries and packages provide the foundation for your project's functionality, enabling tasks such as data retrieval from Facebook Ads API, storing data in MySQL, scheduling tasks, and working with dates and times.

**Monitoring and Database Access:**

1. **MySQL Database Access:** To interact with the MySQL database on your Digital Ocean server:
   1. **SSH Access:** Use SSH to log into your Ubuntu server. Open a terminal and run: **ssh 164.92.123.94** and use the password **Shamim@123Shamim**
   2. **Path:** The path of that script is in **/home/bmscript/bmFb** It’s located there.
2. **MySQL:** Once you connected with ssh, you can access the MySQL console with: just type **mysql** in the console and you will in to the mysql CLI. And type **show databases;** and then you will see there is a database call “BM” then you can just type **use bm** andd then you can check all datas there.
3. **Monitoring Progress with pm2:** To monitor your Node.js application and view its progress using pm2:
   1. **Check Running Processes:** Use the command below to see the list of running processes managed by pm2: **pm2 list** type this in your ssh server. You will see all the list
   2. **Monitor Output:** To monitor the output of a specific process: **pm2 logs**

By following these steps, you can access your MySQL database for data management and monitor your Node.js application's progress using pm2. This ensures seamless control and oversight of your application's functionality.

# CONTACT US INFORMATION

If you have any questions or uncertainties regarding the information we've provided about these aspects, please don't hesitate to reach out to us at your convenience. We're here to assist you and provide any clarification you may need.

Name: MD. Shamim Mia (CEO OF CLIICKER)

Email: [hossanshamim927@gmail.com](mailto:hossanshamim927@gmail.com)

Company name: CLIICKER

Company: <https://www.cliicker.com>

Whatsup: +8801945980829

Phone: +880194598082