Smart Ad Distribution Application

Developed by Manus AI - Your intelligent assistant for marketing task distribution.

Overview

This Streamlit application provides a smart solution for distributing marketing tasks and ad budgets to a team of real estate marketers. It integrates with Google Sheets for data management and uses a weighted algorithm to allocate ads based on project priority, demand, excellence, and remaining marketing size.

Features

- **Google Sheets Integration:** Reads and writes data directly from/to your specified Google Sheet.
- **Data Management:** Loads data for employees, projects, regions, developers, unit types, budgets, and distribution logs.
- **Automatic Ad Distribution:**
  - Select an employee and a region.
  - Input the number of ads to distribute.
  - The application calculates importance scores for active projects in the region.
  - Distributes the specified number of ads proportionally based on project scores.
  - Distributes project ads evenly across the unit types within the project.
- **Balance Tracking:**
  - Displays the current ad balance for the selected employee.
  - Updates employee balances, project ad counts, and the global ad budget in Google Sheets after distribution.
- **Logging:** Records every distribution transaction in the `AdsDistributionLog` sheet.
- **Reporting:** Provides a view of the full distribution log with filtering options by employee and region.
- **Balance Initialization:** Allows initializing or updating employee ad balances based on predefined percentages and the global budget.

Google Sheet Structure

The application expects a Google Sheet (default name: `MarketingData`) with the following worksheets and columns:

1. **GlobalBudget**
    - `GlobalAdsBalance`: The total available ad budget.
2. **Employees**
    - `EmployeeID`: Unique identifier for the employee.
    - `EmployeeName`: Name of the employee.
    - `AdsBudgetPercentage`: Percentage of the global budget allocated initially to this employee (e.g., "10%").
3. **EmployeeBalances**
    - `EmployeeID`: Employee identifier.
    - `AdsBalance`: Current available ad balance for the employee.
4. **Projects**
    - `ProjectID`: Unique identifier for the project.
    - `ProjectName`: Name of the project.
    - `DeveloperID`: Identifier for the project developer.
    - `RegionName`: Name of the region the project is in.
    - `UnitTypesInProject`: Comma-separated list of unit types (e.g., "Apt, Studio, Villa").
    - `ProjectOrder`: Priority order (lower number means higher priority).
    - `Req`: Marketing requirement status ("Yes" or "No").
    - `ProjectExcellenceScore`: Score indicating project excellence.
    - `MarketingSize`: Total marketing size/target for the project.
    - `AdsDistributed`: Number of ads already distributed to this project.
5. **Developers**
    - `DeveloperID`: Unique identifier for the developer.
    - `DeveloperName`: Name of the developer.
    - `DeveloperExcellenceScore`: Score indicating developer excellence.
6. **Regions**
    - `RegionID`: Unique identifier for the region.
    - `RegionName`: Name of the region.
7. **UnitTypes**
    - `UnitTypeID`: Unique identifier for the unit type.
    - `UnitTypeName`: Name of the unit type.
8. **AdsDistributionLog**
    - `DistributionID`: Unique identifier for the distribution record.
    - `EmployeeID`: Employee who received the task.

- `ProjectID`: Project targeted.
- `RegionName`: Region targeted.
- `UnitTypeName`: Unit type targeted.
- `AdsAllocated`: Number of ads allocated in this transaction.
- `DistributionDate`: Timestamp of the distribution.

Setup and Installation

1. **Prerequisites:**
   - Python 3.8+
   - Access to a Google Account with Google Sheets API enabled.
   - A Google Sheet structured as described above.
2. **Google Cloud Setup:**
   - Create a Google Cloud Project.
   - Enable the "Google Drive API" and "Google Sheets API".
   - Create a Service Account.
   - Generate JSON key credentials for the Service Account.
   - **Important:** Share your Google Sheet with the Service Account's email address (found in the JSON key file) and grant it "Editor" permissions.
3. **Clone the Repository (or download the files):** ```bash # If using git # git clone # cd

Or place the downloaded files in a directory:

ad_distribution_app/

- app.py

- google_sheets_utils.py

- distribution_logic.py

- requirements.txt (optional, see below)

4. **Create `requirements.txt` (Optional but recommended):** streamlit pandas gspread oauth2client numpy 5. **Install Dependencies:** bash pip install -r requirements.txt

Or manually: pip install streamlit pandas gspread oauth2client numpy

`` 6. **Configure Streamlit Secrets:** * For deployment (e.g., on Streamlit Community Cloud), create a .streamlit/secrets.toml file in your project directory (ensure this file is **not** committed to public repositories). * Add your Google Service Account JSON key content to the secrets.toml` file like this:

```toml
[google_sheets_credentials]
```

```
type = "service_account"
project_id = "your-project-id"
private_key_id = "your-private-key-id"
private_key = "-----BEGIN PRIVATE KEY-----\nYOUR\nPRIVATE\nKEY\n-----END PRIVATE KEY-----\n"
client_email = "your-service-account-email@your-project-id.iam.gserviceaccount.com"
client_id = "your-client-id"
auth_uri = "https://accounts.google.com/o/oauth2/auth"
token_uri = "https://oauth2.googleapis.com/token"
auth_provider_x509_cert_url = "https://www.googleapis.com/oauth2/v1/certs"
client_x509_cert_url = "https://www.googleapis.com/robot/v1/metadata/x509/your-service-account-email%40your-project-id.iam.gserviceaccount.com"
universe_domain = "googleapis.com"
```

- Replace the placeholder values with the actual content from your downloaded JSON key file. Pay special attention to formatting the `private_key` with `\n` for newlines.
- **Update Google Sheet Name (if necessary):**
- Open `app.py`.
- Find the line `GOOGLE_SHEET_NAME = "MarketingData"`.
- Change `"MarketingData"` to the exact name of your Google Sheet if it's different.

Running the Application

1. Navigate to the application directory in your terminal.
2. Run the Streamlit application: `bash streamlit run app.py`
3. The application should open in your default web browser.

Usage

1. **Data Loading:** The application automatically loads data upon startup. Use the "Refresh Data" button to reload.
2. **Initialize Balances:** Use the "Initialize/Update Employee Balances" button in the sidebar (typically done once or when employee percentages change).
3. **Distribute Ads:**
   - Select an employee from the dropdown.
   - Their current balance will be displayed.
   - Select the target region.
   - Enter the number of ads to distribute.
   - Click "Distribute Ads Automatically".

    - The application will calculate the distribution, show a summary, and attempt to update the Google Sheet.
4. **View Reports:** Scroll down to the "Reports" section to view the full distribution log, with options to filter by employee and region.
5. **Download Logs:** Use the download buttons to get CSV files of the last distribution or the filtered report.

Deployment

This application is ready for deployment on platforms supporting Streamlit, such as Streamlit Community Cloud.

1. Ensure your code is in a Git repository (e.g., GitHub).
2. Make sure your `secrets.toml` file is **not** in the repository (add `.streamlit/secrets.toml` to your `.gitignore` file).
3. Deploy using Streamlit Community Cloud, connecting it to your repository.
4. Add your secrets (the content of your `secrets.toml` file) through the Streamlit Cloud deployment settings interface.

Code Structure

- `app.py` : Main Streamlit application file, handles UI and orchestrates calls.
- `google_sheets_utils.py` : Contains functions for authenticating and interacting with the Google Sheets API.
- `distribution_logic.py` : Contains the core algorithm for calculating importance scores and distributing ads.