**Alter Office Analytics - Documentation**

**Table of Contents**

1. Introduction
2. Prerequisites
3. Installation and Setup
4. Uploading Data to MongoDB
5. Running the FastAPI Server
6. API Endpoints
7. Data Processing and Segmentation
8. Model Building and Clustering
9. Conclusion

**1. Introduction**

This document describes the process of setting up and running a marketing analytics system using MongoDB and FastAPI. The system ingests user data from CSV files, processes unique user records, segments repeated email IDs into a separate cohort collection, and provides API endpoints to access this data. Additionally, clustering models are applied to segment users based on demographics and interests.

**2. Prerequisites**

Ensure the following dependencies are installed before proceeding:

* Python 3.8+
* MongoDB (Installed and Running)
* FastAPI
* Uvicorn
* Pandas
* Pymongo
* Scikit-learn
* Matplotlib
* Seaborn

**3. Installation and Setup**

**Step 1: Install Required Libraries**

Run the following command to install all necessary dependencies:

pip install fastapi uvicorn pandas pymongo scikit-learn matplotlib seaborn

**Step 2: Start MongoDB**

Ensure MongoDB is running on your system:

mongod --dbpath /path/to/mongodb/data

**4. Uploading Data to MongoDB**

**Step 1: Create upload\_csv.py**

Create a script upload\_csv.py to ingest CSV data into MongoDB:

import pandas as pd

from pymongo import MongoClient

from datetime import datetime

# MongoDB Connection

MONGO\_URI = "mongodb://localhost:27017/"

client = MongoClient(MONGO\_URI)

db = client["user\_database"]

users\_collection = db["users"]

unique\_collection = db["unique"]

cohort\_collection = db["cohort"]

# Function to Insert CSV Data into 'users' Collection

def insert\_csv\_to\_users(csv\_path):

df = pd.read\_csv(csv\_path)

records = df.to\_dict(orient="records")

formatted\_records = []

for record in records:

if "created\_at" in record and pd.notna(record["created\_at"]):

record["created\_at"] = datetime.strptime(str(record["created\_at"]), "%m/%d/%Y %H:%M")

if "interests" in record and pd.notna(record["interests"]):

record["interests"] = [interest.strip() for interest in record["interests"].split("|")]

formatted\_record = {"data": record}

formatted\_records.append(formatted\_record)

users\_collection.insert\_many(formatted\_records)

print("✅ Data inserted into 'users' collection")

if \_\_name\_\_ == "\_\_main\_\_":

csv\_path = "D:/alter\_office/sample\_user\_data.csv"

insert\_csv\_to\_users(csv\_path)

**Step 2: Run the Script**

python upload\_csv.py

**5. Running the FastAPI Server**

**Step 1: Create server.py**

from fastapi import FastAPI, HTTPException, Query

from pymongo import MongoClient

from datetime import datetime

from typing import Optional, List, Dict

app = FastAPI()

# MongoDB Connection

MONGO\_URI = "mongodb://localhost:27017/"

client = MongoClient(MONGO\_URI)

db = client["user\_database"]

users\_collection = db["users"]

unique\_collection = db["unique"]

cohort\_collection = db["cohort"]

@app.get("/api/health")

def health\_check():

return {"status": "OK"}

if \_\_name\_\_ == "\_\_main\_\_":

import uvicorn

uvicorn.run(app, host="0.0.0.0", port=8000)

**Step 2: Start the Server**

uvicorn server:app --reload

**6. API Endpoints**

**1. Insert Data via API**

* **URL:** POST /api/ingest
* **Request Body:**

{

"data": {

"cookie": "cookie\_id",

"email": "user@example.com",

"phone\_number": "1234567890",

"created\_at": "03/14/2025 12:00",

"location": {"country": "India", "state": "Tamil Nadu", "city": "Chennai"},

"demographics": {"age": 30, "gender": "Male"},

"interests": ["Gaming", "Music"]

}

}

**2. Fetch User Data**

* **URL:** GET /api/user?cookie=cookie\_id
* **Response:** User details from the unique collection.

**3. Fetch Cohort Data**

* **URL:** GET /api/cohort/user?cookie=cookie\_id
* **Response:** Cohort user details.

**7. Data Processing and Segmentation**

* **Raw data** is first stored in the users collection.
* **Unique users** are identified based on cookies and stored in unique.
* **Duplicate email IDs** are stored separately in cohort for segmentation.
* **Real-time data updates** ensure accurate segmentation.

**8. Model Building and Clustering**

**Step 1: Load Data from MongoDB**

import pandas as pd

from pymongo import MongoClient

# Connect to MongoDB

client = MongoClient("mongodb://localhost:27017/")

db = client["user\_database"]

collection = db["unique"]

# Fetch Data

cursor = collection.find({}, {"\_id": 0})

df = pd.DataFrame(list(cursor))

# Convert 'created\_at' to datetime

df['data.created\_at'] = pd.to\_datetime(df['data.created\_at'], errors='coerce')

# Fill missing values

df.fillna({'data.location.city': 'Unknown', 'data.demographics.income': 'Unknown'}, inplace=True)

df.info()

**Step 2: Data Visualization**

import seaborn as sns

import matplotlib.pyplot as plt

# Age Distribution

plt.figure(figsize=(10,5))

sns.histplot(df['data.demographics.age'].dropna(), bins=20, kde=True, color='blue')

plt.title("Age Distribution of Users")

plt.xlabel("Age")

plt.ylabel("Count")

plt.show()

**Step 3: Clustering Using K-Means**

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

# Select numeric features

features = df[['data.demographics.age']].dropna()

# Scale Data

scaler = StandardScaler()

scaled\_features = scaler.fit\_transform(features)

# Apply K-Means Clustering

kmeans = KMeans(n\_clusters=3, random\_state=42)

df.loc[features.index, 'Cluster'] = kmeans.fit\_predict(scaled\_features)

df[['data.demographics.age', 'Cluster']].head()

**Step 4: Visualizing Clusters**

plt.figure(figsize=(10,6))

sns.scatterplot(x=df.loc[features.index, 'data.demographics.age'], y=df.loc[features.index, 'Cluster'], hue=df.loc[features.index, 'Cluster'], palette="Set2")

plt.title("User Clusters Based on Age")

plt.xlabel("Age")

plt.ylabel("Cluster")

plt.xlim(0, 100)

plt.show()

**9. Conclusion**

This documentation provides a comprehensive guide to setting up and running the **Alter Office Analytics** system, including:

* **Data ingestion**
* **API access**
* **User segmentation**
* **Machine learning-based clustering**

This ensures **real-time marketing analytics** using **MongoDB and FastAPI** with an intelligent cohort-based approach.