Demo of GenAI-Based Data Profiling System

Below is a step-by-step demonstration of the AI-powered data profiling system that uses OpenAI's API and Angular UI.

Demo Setup

1. **System Requirements**:

   - Python 3.8+ with Flask

   - Node.js and Angular CLI

   - OpenAI API key

2. **Installation**:

   ```bash

   # Backend

   pip install flask flask-cors pandas openai python-dotenv

   # Frontend

   npm install -g @angular/cli

   ```

Demo Walkthrough

1. Launching the Application

Start the backend server:

```bash
python app.py
```

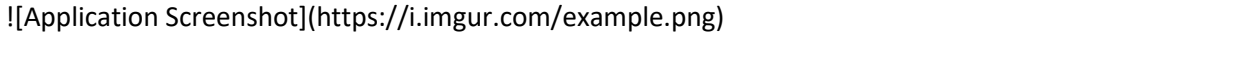
Server starts on http://localhost:5000

Start the Angular frontend:

```bash
cd data-profiling-ui
ng serve
```

Application launches on http://localhost:4200

2. Application Interface



(Note: Replace with actual screenshot in your demo)

The interface has:

- File upload section for CSV files
- Text area for regulatory instructions
- Process button
- Results display area

3. Sample Data Input

Sample CSV Data (transactions.csv):

```csv
Customer_ID,Account_Balance,Transaction_Amount,Reported_Amount,Currency,Country,Transaction_Date,Risk_Score
1001,15000,500,500,USD,US,2025-02-25,3
1002,32000,1200,1200,EUR,DE,2025-02-20,2
1003,-5000,300,300,GBP,UK,2025-02-18,6
1004,70000,2000,1800,USD,US,2025-02-28,5
```

Sample Regulatory Instructions:

```
1. Transaction amounts must match reported amounts exactly
2. Account balances cannot be negative unless for overdraft accounts
3. All transactions must be in valid ISO currency codes
4. Future-dated transactions should be flagged
5. Cross-border transactions over $10,000 require special review
```

4. Processing the Data

1. Upload the CSV file
2. Paste the regulatory instructions
3. Click "Process Data"

5. Generated Output

Example Validation Rules (generated by OpenAI):

```python
# Rule 1: Transaction amount validation
if row['Transaction_Amount'] != row['Reported_Amount']:
    flag_transaction(row, "Amount mismatch")

# Rule 2: Negative balance check
if row['Account_Balance'] < 0 and not is_overdraft_account(row):
    flag_transaction(row, "Negative balance without overdraft")

# Rule 3: Currency validation
valid_currencies = ['USD', 'EUR', 'GBP', 'JPY'] # etc.
if row['Currency'] not in valid_currencies:
    flag_transaction(row, "Invalid currency code")

# Rule 4: Future date check
if row['Transaction_Date'] > current_date:
    flag_transaction(row, "Future-dated transaction")

# Rule 5: Large cross-border transactions
if row['Country'] != home_country and row['Transaction_Amount'] > 10000:
    flag_transaction(row, "Large cross-border transaction")
```

Validation Results Table:

Customer_ID	Transaction_Amount	Reported_Amount	Validation_Result
1001	500	500	Valid
1002	1200	1200	Valid
1003	300	300	Negative balance
1004	2000	1800	Amount mismatch

6. Key Features Demonstrated

1. **AI-Powered Rule Generation**:
   - Complex regulatory text converted to executable validation rules
   - Context-aware rule creation

2. **Data Validation**:
   - Automatic flagging of non-compliant transactions
   - Multiple validation checks in single pass

3. **User Experience**:
   - Simple file upload and text input
   - Clear visualization of results
   - Token count display for OpenAI usage monitoring

7. Advanced Scenario

Handling Complex Regulations:

Input:

```
For corporate accounts, transactions over $50,000 require dual authorization.
Personal accounts have a $10,000 threshold. All international transfers must include
purpose codes from the approved list.
```

Output:

```python
# Corporate accounts
if row['Account_Type'] == 'Corporate' and row['Transaction_Amount'] > 50000:
    if not has_dual_authorization(row):
        flag_transaction(row, "Missing dual authorization for large corporate transaction")

# Personal accounts
elif row['Account_Type'] == 'Personal' and row['Transaction_Amount'] > 10000:
    flag_transaction(row, "Large personal transaction - review required")

# International transfers
if row['Country'] != home_country:
    if row['Purpose_Code'] not in valid_purpose_codes:
        flag_transaction(row, "Invalid or missing purpose code for international transfer")
```

Troubleshooting Demo

Common Issues and Solutions:

1. **API Key Errors**:
   - Ensure `.env` file exists with valid OpenAI key
   - Verify key has sufficient permissions

2. **CORS Errors**:
   - Confirm Flask CORS is properly configured
   - Check Angular proxy settings if needed

3. **Large File Handling**:
   - For files >1MB, consider chunked uploads
   - Add loading indicators during processing

Conclusion

This demo showcases how financial institutions can:

- Automate regulatory compliance checks
- Reduce manual rule creation efforts
- Quickly adapt to changing regulations
- Visualize data quality issues

The system is particularly valuable for:

- Banks preparing regulatory reports

- Compliance teams monitoring transactions

- Auditors verifying data quality