💳 Credit Card Rewards Optimizer: Singapore Edition 🇸🇬

A Singapore-focused application that uses Generative AI to analyze consumer spending patterns and provide multi-card synergy recommendations with strategic usage guidelines. The system uses an agentic architecture leveraging the Model Context Protocol (MCP) to power AI-driven chat for scenario planning and T&C clarifications.

✨ Features

- **Spending Analysis**: Analyze transaction data to create a comprehensive spending profile
- **Merchant Categorization**: Categorize merchant names into spending categories using a distilled model trained on ACRA business data
- **Card Recommendation**: Recommend optimal credit card combinations based on spending patterns and user preferences
- **Strategic Usage**: Provide specific advice on which card to use for which spending category
- **T&C Insights**: Answer natural language questions about card terms and conditions

🌳 Project Structure

```
project/
├── app.py                    # Launcher script for Streamlit app
├── src/                      # Main source code
│   ├── agent/                # Card optimizer agent implementation
│   │   └── agent.py          # Main agent implementation
│   ├── card_processing/      # Card data processing modules
│   │   └── vector_db.py      # Vector database implementation
│   ├── data_collection/      # Data collection utilities
│   ├── model_context_protocol/  # MCP implementation
│   │   ├── client.py         # MCP client for server communication
│   │   └── card_data_server.py  # Card data access server
│   ├── statement_processing/  # Statement analysis modules
│   │   ├── merchant_categorizer.py  # Merchant categorization model
│   │   ├── merchant_categorizer_trainer.py  # Training for categorizer
│   │   └── pdf_statement_parser.py  # PDF parsing module
│   └── user_interface/       # Streamlit UI components
│       ├── Welcome.py        # Main Streamlit entry point
│       ├── components.py     # Reusable UI components
│       ├── utils.py          # UI utility functions
│       └── pages/            # Additional app pages
├── models/                   # Trained models
│   └── merchant_categorizer/  # Merchant categorization models
├── data/                     # Data storage
│   ├── card/                 # Card data
│   ├── card_tcs/             # Card terms and conditions
│   │   └── pdf/              # Original T&C PDFs
│   ├── categorization/       # Categorization data
│   ├── sample_statements/    # Sample card statements for testing
```

```
│   └── vector_db/              # Vector database storage
├── notebooks/                  # Jupyter notebooks for analysis
├── docs/                       # Documentation files
├── results/                    # Analysis results and outputs
├── logs/                       # Application logs
└── requirements.txt            # Python dependencies
```

📋 Requirements

- python>=3.13

🏗️ Setup Instructions

1. Create a virtual environment:

   ```
   python -m venv venv
   source venv/bin/activate  # On Windows: .\venv\Scripts\Activate.ps1
   ```

2. Install dependancies:

   ```
   apt-get install build-essential libpoppler-cpp-dev pkg-config ocrmypdf
   ```

   or

   ```
   brew install gcc@11 pkg-config poppler ocrmypdf
   ```

   For Windows, you have to manually find these binaries and ensure they are discoverable. It is likely that these would be available on Windows package managers like `Winget` or `Chocolatey`, but I have not tested them yet.

3. Install requirements:

   ```
   pip install -r requirements.txt
   ```

4. Create a .env file by copying .env.example and filling in required values.

5. Start the MCP server:

   ```
   python -m src.model_context_protocol.card_data_server
   ```

6. (**In a new terminal window**) Run the Streamlit application:

```
python app.py
```

For Docker, see DOCKER.md.

🧐 Component Overview

1. Transaction Categorization

- `src/statement_processing/merchant_categorizer.py` - Model for categorizing merchant names
- `src/statement_processing/merchant_categorizer_trainer.py` - Training pipeline for categorizer model
- `src/statement_processing/pdf_statement_parser.py` - PDF statement parsing and data extraction

2. Card Embeddings & Semantic Search

- `src/card_processing/vector_db.py` - Vector database for card embeddings and T&C documents
- `src/model_context_protocol/card_data_server.py` - MCP tools for card data access including semantic search

3. Agent Reasoning

- `src/agent/agent.py` - Main agent implementation for card recommendations and scenario analysis

4. RAG Pipeline

- `src/card_processing/vector_db.py` - Vector database for T&C document storage
- `src/model_context_protocol/card_data_server.py` - Tools for T&C querying

5. UI & Application Flow

- `app.py` - Main application entry point
- `src/user_interface/Welcome.py` - Streamlit UI main page
- `src/user_interface/components.py` - Reusable UI components
- `src/user_interface/utils.py` - UI utility functions

MCP Tool Architecture

The system implements core MCP tools through the `src/model_context_protocol/card_data_server.py`:

1. `get_available_cards()` - Returns a list of all available cards with basic metadata
2. `get_card_details(card_id)` - Returns complete card information in its original format
3. `query_tc(question, card_id)` - Natural language queries about card terms and conditions
4. `search_cards(query)` - Semantic search for cards matching natural language criteria