Qirat AI Smart Learning API

هذا المشروع يوفر واجهة برمجية (API) لتحليل الأهداف والمعاملات المالية بناءً على قواعد محددة، مع إضافة ميزة **التعلم المستمر من سلوك المستخدم** و**محرك التحفيز الذكي**، وذلك لدعم تطبيق قيراط في تقديم توصيات ذكية ومخصصة للمستخدمين.

الميزات

- تحليل الأهداف المالية وتقديم تنبيهات حول المهلة الزمنية ومعدل التقدم، بالإضافة إلى تحليل المبلغ اليومي المطلوب.
- تحليل المعاملات المالية وتقديم تحذيرات حول تجاوز الميزانية، تنبيهات الاقتراب من الحد، تحليل سرعة الإنفاق، واقتراحات لتحسين الإنفاق بناءً على فئات الرغبات.
- **التعلم المستمر**: يقوم النظام الآن بتعديل أولوية التوصيات بناءً على تفاعلات المستخدم السابقة (قبول، رفض، تجاهل).
- **محرك التحفيز الذكي**: يكتشف الإنجازات المالية للمستخدم (مثل الاقتراب من تحقيق هدف أو الانضباط في الميزانية أو التوفير الاستثنائي) ويقدم رسائل تشجيعية مخصصة.
- واجهة برمجة تطبيقات RESTful سهلة الاستخدام.

المتطلبات

- Python 3.8+
- pip (مدير الحزم لـ Python)
- FastAPI, Uvicorn, Pydantic
- SQLite3 (مضمن مع Python)

الإعداد والتشغيل

اتبع الخطوات التالية لإعداد وتشغيل الـ API:

1. **استنساخ المشروع (أو إنشاء الملفات يدوياً):**

إذا كنت قد استلمت الملفات مباشرة، فتأكد من أنها موجودة في مجلد واحد (مثل `qirat_ai_api`).

2. **الانتقال إلى مجلد المشروع:**

```bash
cd /home/ubuntu/qirat_ai_api
```

3. **تثبيت التبعيات:**

قم بتثبيت الحزم المطلوبة باستخدام `pip` :

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

4. **تهيئة قاعدة البيانات:**

تأكد من تهيئة قاعدة البيانات SQLite التي ستخزن تفاعلات المستخدمين:

```bash
python3 database.py
```

5. **تشغيل الـ API:**

يمكنك تشغيل الخادم باستخدام Uvicorn:

```bash
uvicorn main:app --host 0.0.0.0 --port 8000
```

سيتم تشغيل الـ API على العنوان `http://0.0.0.0:8000` . يمكنك الوصول إلى وثائق Swagger UI التفاعلية على `http://0.0.0.0:8000/docs` .

نقاط النهاية (API Endpoints)

1. تحليل الأهداف المالية

- **المسار:** `analyze/goal/`
- **الطريقة:** `POST`
- **الوصف:** يقوم بتحليل هدف مالي محدد وتقديم توصيات بناءً على حالته، مع الأخذ في الاعتبار الأوزان المتعلمة من سلوك المستخدم، بالإضافة إلى رسائل تشجيعية عند الاقتراب من تحقيق الهدف.
- **نموذج الطلب (Request Body):**

```json
{
    "id": "goal_123",
    "user_id": "user_abc",
    "title": "شراء سيارة",
    "target_amount": 50000.0,
    "current_amount": 25000.0,
```

```
    "deadline": "2026-12-31"
}
```

- **نموذج الاستجابة (Response Body):**

JSON

```
{
    "feedbacks": [
        {
            "id": "<unique_feedback_id>",
            "type": "info",
            "message": " معدل التقدم منخفض (%50.0). قد تحتاج لزيادة مدخراتك لتجنب التأخير.",
            "action_type": "إضافة مبلغ",
            "priority": 2,
            "score": 1.0
        }
    ]
}
```

2. تحليل المعاملات المالية

- **المسار:** `analyze/transaction/`
- **الطريقة:** `POST`
- **الوصف:** يقوم بتحليل معاملة مالية وتقديم تنبيهات أو اقتراحات، مع الأخذ في الاعتبار الأوزان المتعلمة من سلوك المستخدم، بالإضافة إلى رسائل تشجيعية عند الانضباط المالي أو التوفير الاستثنائي.
- **نموذج الطلب (Request Body):**

JSON

```
{
    "id": "trans_456",
    "user_id": "user_abc",
    "amount": 1200.0,
    "category": "ترفيه",
    "description": "عشاء في مطعم فاخر",
    "date": "2026-01-10",
    "budget_limit": 1000.0
}
```

- **نموذج الاستجابة (Response Body):**

JSON

```
{
    "feedbacks": [
        {
            "id": "<unique_feedback_id>",
            "type": "warning",
            "message": "تنبيه: هذه المعاملة (1200.0) تتجاوز حد الميزانية المسموح به
(1000.0).",
            "action_type": null,
            "priority": 1,
            "score": 1.0
        },
        {
            "id": "<unique_feedback_id>",
            "type": "info",
            "message": "هذه المعاملة تندرج تحت \"الرغبات\". تقليل الإنفاق في فئة
يساعدك على تحقيق أهدافك أسرع "\ترفيه"\.",
            "action_type": null,
            "priority": 4,
            "score": 1.0
        }
    ]
}
```

3. إرسال التغذية الراجعة (Feedback)

- **المسار:** `feedback/`
- **الطريقة:** `POST`
- **الوصف:** يسمح لتطبيق العميل بإرسال تفاعلات المستخدم مع التوصيات، مما يساعد النظام على التعلم وتعديل أوزان التوصيات المستقبلية.
- **نموذج الطلب (Request Body):**

JSON

```
{
    "user_id": "user_abc",
    "feedback_id": "deadline_warning", # نوع التوصية التي تم التفاعل معها
    "action": "accepted" # أو "dismissed" أو "ignored"
}
```

- **نموذج الاستجابة (Response Body):**

JSON

```
{
    "status": "success",
    "message": "Feedback recorded, model is learning..."
}
```

هيكل المشروع

- `main.py` : نقطة الدخول الرئيسية لتطبيق FastAPI، ويحتوي على تعريف نقاط النهاية.
- `models.py` : يحدد نماذج البيانات (Goal, Transaction, FeedbackItem, AnalysisResult, UserFeedback) باستخدام Pydantic.
- `analyzer.py` : يحتوي على منطق التحليل المالي الفعلي (AIFinancialAnalyzer)، بناءً على القواعد، بالإضافة إلى منطق الرسائل التشجيعية. ، `database.py` ويستخدم الأوزان المتعلمة من
- `database.py` : يدير قاعدة بيانات SQLite لتخزين تفاعلات المستخدمين وأوزان التوصيات.
- `requirements.txt` : يسرد جميع التبعيات المطلوبة لتشغيل المشروع.

ملاحظات هامة

- هذا النموذج يجمع بين القواعد (Rule-based) والتعلم البسيط من التغذية الراجعة. يمكن تطويره مستقبلاً ليشمل نماذج تعلم آلي أكثر تعقيداً.
- تاريخ اليوم (`date.today()`) يستخدم في التحليل، لذا تأكد من ضبط توقيت الخادم بشكل صحيح.
- يمكنك تعديل القواعد في `analyzer.py` لتناسب احتياجاتك بشكل أفضل.
- لتبسيط عملية التعلم، تم استخدام `feedback_id` في `UserFeedback` لتمثيل `feedback_type` مباشرة. في نظام إنتاجي أكبر، قد تحتاج إلى ربط `feedback_id` بمعرف فريد لكل توصية تم إنشاؤها وتخزينها في قاعدة البيانات.

المؤلف: Manus AI

التاريخ: 11 يناير 2026