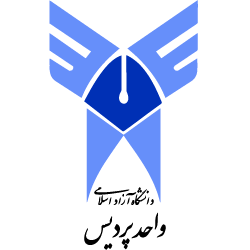
بسمه تعالی



**مستندات درس پروژه نرم‌افزار**

دانشگاه آزاد اسلامی واحد پردیس

استاد: خانم نسیم وطنی  
دانشجو: ارمیا شعبانی

نیمسال دوم (بهمن ماه) ۱۴۰۳

فهرست مطالب

 **مقدمه**  
 انگیزهٔ ایجاد پروژه  
 دامنهٔ پروژه  
 مخاطبان هدف

 **مرور کلی سیستم**  
 معماری کلی  
 جریان داده‌ها  
 نمودار سطح بالا

 **فناوری‌ها و ابزارها**  
 زبان برنامه‌نویسی و فریم‌ورک‌ها  
 پایگاه داده (PostgreSQL)  
 سرویس‌های خارجی (CoinDesk, CoinGecko, Messari)  
 مدیریت بسته‌ها و محیط مجازی

 **طراحی و ساختار پروژه**  
 ساختار دایرکتوری  
 توضیح فایل‌های اصلی main.py, قالب‌ها، استاتیک  
 مدیریت پیکربندی و متغیرهای محیطی

 **جزئیات پیاده‌سازی بک‌اند**  
 راه‌اندازی و اتصال به پایگاه داده  
 کلاس CryptoNewsTracker و متدهای کلیدی  
 واکشی اخبار (Fetch News)  
 تشخیص نام رمزارزها در متن (Coin Mentioning)  
 واکشی قیمت (Current & Historical Price)  
 ذخیره‌سازی در دیتابیس و روابط جداول

 **API و نقاط پایانی (Endpoints)**  
 REST API   
 توضیح پارامترهای ورودی و خروجی  
 مثالی از درخواست و پاسخ JSON

 **جزئیات پیاده‌سازی فرانت‌اند**  
 قالب‌های HTML (index, post)  
 استایل‌ها و CSS Tailwind،Loader   
 اسکریپت‌های JavaScript  
 واکنش‌گرایی (Responsive Design)  
 تعاملات کاربر (فیلتر، جستجو، پشتیبانی از تاریخ)

 **مدیریت کش و بهینه‌سازی عملکرد**  
 کش قیمت‌ها در حافظه  
 ایجاد ایندکس‌ها در PostgreSQL  
 جلوگیری از اخبار تکراری  
 بهبود زمان پاسخ API

 **ملاحظات امنیتی**  
 نگهداری ایمن کلیدهای API  
 اعتبارسنجی و پاک‌سازی ورودی‌ها  
 محدود کردن نرخ درخواست‌ها  
 تنظیم CORS و HTTPS

 **آزمون و تضمین کیفیت**  
 تست‌های واحد (Unit Tests)  
 تست‌های یکپارچه‌سازی (Integration Tests)  
 تست دستی فرانت‌اند  
 معیارهای پذیرش

 **راهنمای راه‌اندازی و استقرار**  
 نصب پیش‌نیازها  
 پیکربندی محیط (Virtualenv, .env)  
 اجرای محلی  
 استقرار روی سرور / Docker

 **مدیریت عملیات و نگهداری**  
 نظارت بر عملکرد (Logging, Monitoring)  
 بک‌آپ و بازیابی پایه‌داده  
 به‌روزرسانی و نگهداری دوره‌ای

 **نقاط قابل بهبود و توسعه‌های آینده**  
 اضافه کردن منابع خبری بیشتر  
 پشتیبانی از رابط کاربری پیشرفته  
 یادگیری ماشین برای تحلیل احساسات

 **نتیجه‌گیری**  
 دستاوردها  
 چالش‌ها و تجربیات  
 چشم‌انداز

 **ضمائم**  
 فهرست واژگان  
 ساختار کامل بانک اطلاعاتی (ER Diagram)  
 نمونه کدهای کلیدی  
 منابع و مآخذ

**مقدمه**

**انگیزهٔ ایجاد پروژه**

در سال‌های اخیر بازار ارزهای دیجیتال با رشد چشمگیری مواجه شده و حجم عظیمی از اخبار، تحلیل‌ها و داده‌های قیمتی در فضای وب منتشر می‌شود. برای سرمایه‌گذاران و علاقه‌مندان، پیگیری هم‌زمان اخبار متعدد از منابع گوناگون و تحلیل تأثیرات آن‌ها بر قیمت لحظه‌ای رمزارزها کاری دشوار و زمان‌بر است. از سوی دیگر، بسیاری از پلتفرم‌های خبری، تنها متن خبر را منتشر می‌کنند و ارتباط مستقیم میان خبر و وضعیت قیمتی در لحظهٔ انتشار یا اکنون را در اختیار کاربر نمی‌گذارند.  
  
لذا در این پروژه بر آن شدیم سامانه‌ای طراحی کنیم که:

* اخبار مهم حوزهٔ کریپتوکارنسی را به‌صورت خودکار از منابع معتبر جمع‌آوری کند،
* علاوه بر متن خبر، اطلاعات قیمتی لحظه‌ای و قیمت در زمان انتشار را نیز ثبت کرده و نمایش دهد،
* با ذخیره‌سازی و نگاشت مناسب بین خبر و رمزارزهای نام‌برده، تحلیل ترکیبی خبر و قیمت را برای کاربر تسهیل نماید.

**دامنهٔ پروژه**

هدف اصلی این سامانه، فراهم کردن بستری جامع برای:

1. **گردآوری اخبار** از چندین منبع شناخته‌شده مانند CoinDesk و CryptoCompare،
2. **پیگیری قیمت رمزارزها** در دو بازهٔ زمانی—قیمت لحظه‌ای و قیمت در زمان انتشار خبر—
3. **ارتباط‌سازی داده‌ها** به گونه‌ای که کاربر بتواند به‌راحتی ببیند کدام رمزارزها در یک خبر خاص مطرح شده و قیمت آن‌ها در دو زمان متفاوت چه تغییراتی داشته است.

دامنه این پروژه شامل موارد زیر است:

* توسعهٔ بک‌اند با زبان پایتون و فریم‌ورک Flask
* طراحی پایگاه داده PostgreSQL برای ذخیره اخبار، قیمت‌ها و روابط
* ساخت رابط وب ساده اما کاربردی با HTML/CSS/JavaScript و Tailwind
* پیاده‌سازی REST API برای دسترسی برنامه‌ای به اخبار و قیمت‌ها
* بهینه‌سازی کارایی از طریق کشینگ داده‌ها و ایندکس‌گذاری

**مخاطبان هدف**

این سامانه می‌تواند برای گروه‌های زیر مفید باشد:

* **سرمایه‌گذاران فردی**: کسانی که می‌خواهند با بررسی ترکیبی خبر و نوسان قیمت در زمان انتشار، تصمیمات بهتری بگیرند.
* **تحلیل‌گران مالی**: افرادی که نیاز دارند رخدادهای خبری و تأثیر آن‌ها بر بازار را به‌صورت داده‌محور تحلیل کنند.
* **توسعه‌دهندگان و پژوهشگران حوزه بلاک‌چین**: برای استفاده از API پروژه در کارهای تحقیقاتی یا یکپارچه‌سازی با سایر ابزارها.
* **علاقه‌مندان و دانشجویان**: کسانی که می‌خواهند ساختار یک پروژهٔ کامل Flask + PostgreSQL را به‌عنوان نمونه ببینند و روی آن مطالعه یا توسعه دهند.

**مرور کلی سیستم**

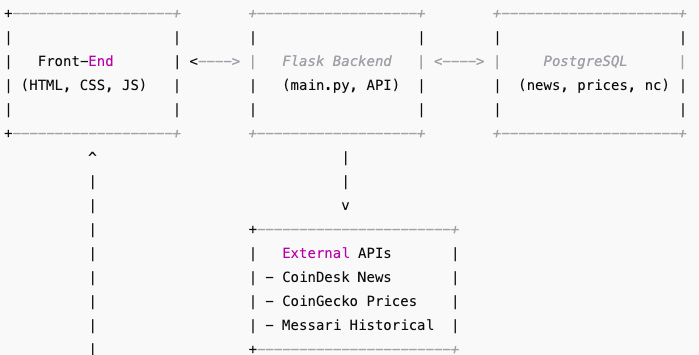
**معماری کلی**

سامانه‌ی Crypto News Tracker از سه لایه‌ی اصلی تشکیل شده است:

1. **لایهٔ دیتابیس (Persistence Layer)**
   * PostgreSQL به عنوان پایگاه دادهٔ رابطه‌ای برای ذخیره‌سازی اخبار (crypto\_news)، قیمت‌های رمزارز (crypto\_prices) و نگاشت بین خبر و رمزارز (news\_coins)
   * ایندکس‌گذاری در جداول برای جستجوی سریع بر اساس تاریخ انتشار و متن
2. **لایهٔ بک‌اند (Application Layer)**
   * یک برنامهٔ Flask در پایتون که شامل:
     + ماژول‌های اتصال و مدیریت پایگاه داده
     + کلاس CryptoNewsTracker برای گردآوری و پردازش اخبار و قیمت‌ها
     + متدهای واکشی اخبار از APIهای خارجی، تشخیص نام رمزارزها، کشینگ قیمت و ذخیره در دیتابیس
     + نقطه‌پایانی (Endpoints) وب برای نمایش صفحات HTML و REST API برای دسترسی برنامه‌ای
3. **لایهٔ فرانت‌اند (Presentation Layer)**
   * قالب‌های HTML (بسته به Tailwind CSS)
   * فایل‌های CSS برای استایل و انیمیشن لودر
   * JavaScript برای:
     + فراخوانی REST API (اخبار، فهرست رمزارزها، قیمت‌ها)
     + فیلتر و جستجوی اخبار
     + نمایش قیمت لحظه‌ای و قیمت زمان خبر به‌صورت واکنش‌گرا
     + مدیریت حالت بارگذاری و صفحه‌آرایی

**جریان داده‌ها**

1. **شروع کار برنامه**
   * اسکریپت main.py با فراخوانی create\_tables() و recreate\_indexes() مطمئن می‌شود ساختار دیتابیس آماده است.
   * سپس CryptoNewsTracker.run() در یک ترد جداگانه اجرا می‌شود تا به‌طور مستمر اخبار را از APIها دریافت کند.
2. **واکنش بک‌اند برای APIها**
   * کاربر یا فرانت‌اند با درخواست HTTP به /api/news، /api/coins یا /api/prices، داده‌ها را از دیتابیس یا کش دریافت کرده و JSON برمی‌گرداند.
   * در نقطه‌پایانی‌های وب (/ و /post/<id>) قالب‌های HTML رندر شده و به مرورگر ارسال می‌شوند.
3. **جمع‌آوری اخبار و قیمت**
   * در حلقهٔ اصلی CryptoNewsTracker.run():
     1. فراخوانی fetch\_news() برای دریافت لیست اخبار جدید (حداکثر ۱۰۰ خبر) از CoinDesk
     2. برای هر خبر، با check\_coin\_mentions() نام رمزارزهای مطرح‌شده استخراج می‌شود.
     3. متد store\_news() خبر و قیمت لحظه‌ای (از CoinGecko) و قیمت تاریخی (از Messari یا CoinGecko) را در جداول مربوط ذخیره می‌کند.
4. **نمایش در فرانت‌اند**
   * JavaScript:
     1. ابتدا /api/coins را برای پر کردن منوی فیلتر رمزارزها فرا می‌خواند.
     2. سپس با پارامترهای فیلتر، /api/news را صدا می‌زند و نتایج را داخل کارت‌های خبری نمایش می‌دهد.
     3. هر کارت شامل برند تاریخ/زمان، پیش‌نمایش متن خبر و برچسب‌های رمزارز است که با حرکت ماوس قیمت تاریخی و لحظه‌ای نمایش می‌دهد.
     4. در ستون کناری (سایدبار) با فراخوانی /api/prices قیمت‌های لحظه‌ای کلیه رمزارزها را به‌روزرسانی می‌کند.

**نمودار سطح بالا**

* ارتباط‌ها:
  + **Front-End ⇄ Flask Backend**:
    - بارگذاری صفحات HTML
    - فراخوانی REST API
  + **Flask Backend ⇄ PostgreSQL**:
    - خواندن و نوشتن اخبار و قیمت‌ها
  + **Flask Backend ⇄ External APIs**:
    - واکشی اخبار و داده‌های قیمتی

**فناوری‌ها و ابزارها**

**زبان برنامه‌نویسی و فریم‌ورک‌ها**

* **پایتون (Python)**  
  زبان اصلی پروژه برای پیاده‌سازی بک‌اند و منطق کسب‌وکار است. انتخاب پایتون به دلیل سادگی نحو، کتابخانه‌های غنی برای کار با وب و داده و اکوسیستم فعال آن مناسب بود.
* **فلسک (Flask)**  
  فریم‌ورک سبک و منعطف وب برای ساخت REST API و رندر قالب‌ها. فلسک با داشتن قابلیت‌های میدل‌ور سفارشی، مدیریت آسان مسیریابی و پلاگین‌های گسترده، گزینه‌ای ایده‌آل برای این پروژه محسوب می‌شود.

**پایگاه داده (PostgreSQL)**

* **PostgreSQL**  
  یک سامانهٔ مدیریت پایگاه داده رابطه‌ای متن‌باز (RDBMS) قدرتمند با پشتیبانی از داده‌های پیچیده و افزوده‌پذیری بالا.
  + جداول اصلی:
    - crypto\_news برای ذخیره اطلاعات مقالات خبری
    - crypto\_prices برای ثبت قیمت لحظه‌ای و تاریخی رمزارزها
    - news\_coins برای نگاشت بین خبر و رمزارز
  + امکانات کلیدی به‌کار رفته:
    - ایندکس‌گذاری با GIN برای جستجوی متنی در عنوان
    - ایندکس روی ستون published\_date برای مرتب‌سازی سریع بر اساس تاریخ
    - قیدهای یکتا برای جلوگیری از ذخیرهٔ اخبار تکراری

**سرویس‌های خارجی**

1. **CoinDesk API**
   * منبع اصلی دریافت اخبار کریپتوکارنسی.
   * برای احراز هویت نیاز به کلید API (COINDESK\_API\_KEY) دارد.
2. **CoinGecko API**
   * ارائه‌دهندهٔ داده‌های لحظه‌ای قیمت و حجم معاملاتی رمزارزها.
   * نیازی به کلید API ندارد، اما برای افزایش نرخ درخواست می‌توان از برنامهٔ رایگان یا پولی آن بهره برد.
3. **Messari API**
   * برای دریافت قیمت تاریخی دقیق بر اساس بازهٔ زمانی خاص.
   * استفاده از کلید API (MESSARI\_API\_KEY) اختیاری است، اما نرخ محدودیت کمتری اعمال می‌کند.

**مدیریت بسته‌ها و محیط مجازی**

* **Virtualenv / venv**
  + برای جداسازی وابستگی‌های پروژه و جلوگیری از تداخل نسخه‌ها از محیط مجازی پایتون استفاده می‌شود.
* **pip & requirements.txt**
  + مدیریت بسته‌ها با pip و فهرست شدن نسخه‌های موردنیاز در فایل requirements.txt (مانند Flask, psycopg2, requests, python-dotenv).
* **dotenv**
  + بارگیری متغیرهای محیطی از فایل .env برای پیکربندی امن کلیدهای API و پارامترهای پایگاه داده.

**طراحی و ساختار پروژه**

**ساختار دایرکتوری  
  
**

* دایرکتوری static/: شامل فایل‌های CSS، تصاویر آیکون رمزارزها و انیمیشن لودر
* دایرکتوری templates/: شامل دو فایل قالب با استفاده از Jinja2 برای رندر صفحات
* فایل main.py: محتوا و توضیح در ادامه

**توضیح فایل‌های اصلی**

* **main.py**
  + بارگذاری متغیرهای محیطی (dotenv) و پیکربندی Flask
  + تعریف کلاس CryptoNewsTracker به‌همراه متدهای:
    - fetch\_news()
    - check\_coin\_mentions()
    - get\_coin\_price() و get\_historical\_price()
    - store\_news()
    - run() برای اجرای مداوم در ترد جداگانه
  + تعریف توابع کمکی برای ساخت جداول (create\_tables) و ایندکس‌ها (recreate\_indexes)
  + تعریف نقاط پایانی وب و REST API
  + اجرای همزمان اسکریپت جمع‌آوری اخبار و سرور Flask
* **requirements.txt**
  + فهرست نسخه‌های مورد نیاز کتابخانه‌ها مانند Flask==2.x, psycopg2-binary, requests, python-dotenv
* **.env**
  + ذخیرهٔ امن مقادیر:

DB\_NAME=…

DB\_USER=…

DB\_PASSWORD=…

DB\_HOST=…

DB\_PORT=…

COINDESK\_API\_KEY=…

MESSARI\_API\_KEY=…

* **DOCUMENTATION.md و مستندات.md**
  + مستندات اولیه به زبان فارسی و انگلیسی؛ شامل فهرست مطالب، راهنما و طراحی پروژه
* **README.md**
  + راهنمای نصب و اجرای سریع پروژه برای کاربران

**مدیریت پیکربندی و متغیرهای محیطی**

1. **بارگذاری فایل .env**  
   با استفاده از بستهٔ python-dotenv در ابتدای main.py فراخوانی می‌شود:

from dotenv import load\_dotenv

load\_dotenv(override=True)

1. **دریافت مقادیر محیطی**

DB\_CONFIG = {

"dbname": os.getenv('DB\_NAME'),

"user": os.getenv('DB\_USER'),

"password": os.getenv('DB\_PASSWORD'),

"host": os.getenv('DB\_HOST'),

"port": os.getenv('DB\_PORT')

}

COINDESK\_API\_KEY = os.getenv('COINDESK\_API\_KEY')

MESSARI\_API\_KEY = os.getenv('MESSARI\_API\_KEY', '')

1. **امنیت و گیت‌ایدیگنور**
   * فایل .env در .gitignore قرار دارد تا مقادیر حساس مخزن نشود.
   * کلیدهای API و اطلاعات دیتابیس فقط در محیط محلی یا سرور ذخیره می‌شوند.

**جزئیات پیاده‌سازی بک‌اند**

**راه‌اندازی و اتصال به پایگاه داده**

1. **تابع get\_db\_connection()**

def get\_db\_connection():

conn = psycopg2.connect(\*\*DB\_CONFIG)

conn.set\_client\_encoding('UTF8')

conn.autocommit = True

return conn

* + وظیفهٔ ایجاد و بازگرداندن یک اتصال جدید به PostgreSQL را دارد.
  + autocommit=True برای خودکار کردن تراکنش‌ها و جلوگیری از نیاز صریح به commit() در توابع سادهٔ خواندنی به‌کار رفته است.

1. **متدهای connect\_db() و close\_db() در کلاس CryptoNewsTracker**
   * connect\_db() مقداردهی اولیهٔ اتصال و کرسر را انجام می‌دهد و خطاهای اتصال را مدیریت می‌کند.
   * close\_db() کرسر و اتصال را می‌بندد تا نشتی منابع رخ ندهد.

**کلاس CryptoNewsTracker و متدهای کلیدی**

این کلاس هستهٔ عملیات جمع‌آوری و ذخیره‌‌سازی اخبار و قیمت‌هاست.

* **سازنده (\_\_init\_\_)**: پارامتر db\_config را می‌گیرد و متغیرهای داخلی را مقداردهی می‌کند.
* **متد run()**:
  1. اتصال به دیتابیس
  2. فراخوانی مداوم fetch\_news() برای دریافت اخبار
  3. برای هر خبر: تشخیص رمزارزهای نام‌برده و ذخیره با store\_news()
  4. وقفهٔ یک ساعته قبل از تکرار
* **متد fetch\_news(start\_date, end\_date)**:
  1. ارسال درخواست به CoinDesk برای دریافت اخبار اخیر (حداکثر ۱۰۰ مورد)
  2. فیلتر بر اساس بازهٔ زمانی
  3. برگرداندن لیست دیکشنری‌های هر خبر

**واکشی اخبار (Fetch News)**

* **مسیر دسترسی و هدرها**  
  از NEWS\_SOURCES['coindesk']['url'] با هدر احراز هویت استفاده می‌شود.
* **پردازش پاسخ**
  + JSON دریافتی در کلید Data لیست اخبار دارد.
  + با گام‌ (batch) ۲۰تایی پردازش می‌شود تا عملکرد بهتری فراهم آید.
* **فیلتر زمانی**  
  اگر published\_on هر خبر بین start\_date و end\_date باشد، در خروجی قرار می‌گیرد.

**تشخیص نام رمزارزها در متن (Coin Mentioning)**

* متد check\_coin\_mentions(article: Dict) -> Set[str]
  + متن عنوان و بدنهٔ خبر به حروف کوچک تبدیل و نویسه‌های جداکننده حذف می‌شود.
  + یک دیکشنری از نام‌ها و نمادهای هر رمزارز تعریف شده:

coins = {

"bitcoin": ["bitcoin", "btc"],

"ethereum": ["ethereum", "eth"],

...

}

* + با جستجوی کلمه به‌صورت مرزبندی‌شده (f" {name} ") هر بار مشاهده، نام کلید (مثل ("bitcoin" به مجموعه اضافه می‌شود.

**واکشی قیمت (Current & Historical Price)**

1. **قیمت لحظه‌ای با CoinGecko**
   * متد get\_coin\_price(coin\_id) پارامترها را به /simple/price می‌فرستد و دادهٔ JSON شامل قیمت، ارزش بازار و حجم ۲۴ ساعته برمی‌گردد.
   * با قفل (price\_cache\_lock) و دکوریتور @lru\_cache روی get\_cached\_price از فراخوانی‌های مکرر جلوگیری می‌شود (کش ۵ دقیقه‌ای(.
2. **قیمت تاریخی با Messari**
   * متد get\_historical\_price(coin\_id, timestamp) درخواست به /metrics/price/time-series ارسال می‌کند.
   * نتایج شامل آرایهٔ [timestamp, price] است که قیمت بسته شدن آن روز را برمی‌گرداند.

**ذخیره‌سازی در دیتابیس و روابط جداول**

* **متد store\_news(article, mentioned\_coins)**
  1. بررسی وجود URL در جدول crypto\_news
     + اگر وجود داشت و تاریخ انتشار جدیدتر است، با UPDATE به‌روزرسانی می‌شود.
     + در غیر این صورت با INSERT یک رکورد جدید ایجاد می‌شود.
  2. برای هر رمزارز ذکرشده:
     + دریافت قیمت لحظه‌ای و تاریخی
     + ذخیرهٔ رکورد جدید در crypto\_prices
     + ایجاد رابطه در news\_coins با قیمت در زمان خبر
  3. تراکنش با commit() پایدار می‌شود و در صورت خطا به rollback() بازمی‌گردد.

**API و نقاط پایانی (Endpoints)**

**صفحات وب (Web Endpoints)**

**مثال درخواست و پاسخ**

GET /post/42 HTTP/1.1

Host: localhost:5000

* پاسخ: صفحه‌ی HTML شامل عنوان، محتوا، تاریخ و منبع خبر با قالب‌بندی مناسب.

**REST API عمومی**

**لیست اخبار**

* **مسیر:** /api/news
* **روش:** GET
* **پارامترها (کوئری‌استرینگ:(**
  + coin )اختیاری): فیلتر بر اساس شناسه‌ی رمزارز (مثلاً bitcoin، (eth
  + search )اختیاری): جستجو در عنوان خبر (عبارت متنی(
  + date\_from )اختیاری): حد پایین تاریخ انتشار به فرمت YYYY-MM-DD یا YYYY-MM-DD HH:MM
  + date\_to )اختیاری): حد بالای تاریخ انتشار
  + limit )اختیاری): تعداد نتایج بازگشتی (مقدار پیش‌فرض (10
  + offset )اختیاری): جابجایی برای نمایش صفحه‌ای (پیش‌فرض (0
* **پاسخ (JSON):**

[

{

"id": 42,

"title": "Bitcoin Hits New All-Time High",

"content": "The price of Bitcoin reached...",

"published\_date": "2025-07-15T14:23:00",

"source": "CoinDesk",

"url": "https://www.coindesk.com/...",

"coins": ["bitcoin", "ethereum"],

"prices": {

"bitcoin": {

"historical": {"price": 40000.12},

"current": {"price": 41000.55}

},

"ethereum": {

"historical": {"price": 2300.45},

"current": {"price": 2350.78}

}

}

},

...

]

* **توضیح فیلدها:**
  + coins: آرایه‌ای از شناسه رمزارزهای ذکرشده.
  + prices: شیئی از شناسه‌ی رمزارز به دو زیرشیء:
    - historical.price: قیمت رمزارز هنگام انتشار مقاله.
    - current.price: آخرین قیمت ثبت شده (لحظه‌ای).

**فهرست رمزارزها**

* **مسیر:** /api/coins
* **روش:** GET
* **پاسخ (JSON):**

["bitcoin", "ethereum", "ripple", "dogecoin", "solana", "tron", "tether"]

**6.2.3 قیمت‌های لحظه‌ای**

* **مسیر:** /api/prices
* **روش:** GET
* **توضیح عملکرد:**
  1. تلاش برای دریافت قیمت‌های تازه از جدول ) crypto\_prices داده‌های کمتر از ۵ دقیقه قبل.(
  2. در صورت ناکافی‌بودن داده‌های تازه، واکشی مجدد از CoinGecko و ذخیره در دیتابیس.
* **پاسخ (JSON):**

[

["bitcoin", 41000.55],

["ethereum", 2350.78],

["solana", 50.12],

...

]

* هر عنصر آرایه شامل [coin\_id, price\_usd] است و بر اساس قیمت نزولی مرتب شده.

**نمونه درخواست و پاسخ با فیلتر**

**درخواست:**

GET /api/news?coin=bitcoin&date\_from=2025-07-01&limit=5 HTTP/1.1

Host: localhost:5000

**عملکرد:**

* بازگرداندن ۵ خبر جدید حاوی «bitcoin» که از اول ژوئیهٔ ۲۰۲۵ یا بعد آن منتشر شده‌اند.

**نمونه پاسخ:**

[

{

"id": 55,

"title": "Bitcoin ETF Approved by SEC",

"published\_date": "2025-07-20T10:15:00",

"coins": ["bitcoin"],

"prices": {

"bitcoin": {

"historical": {"price": 45000.00},

"current": {"price": 45500.20}

}

}

},

...

]

**جزئیات پیاده‌سازی فرانت‌اند**

**قالب‌های HTML**

* **templates/index.html**  
  صفحهٔ اصلی که شامل:
  1. **ساختار اسکلت**: نوار کناری (Sidebar) و بخش محتوای اصلی (News Grid)
  2. **المان‌های فرم فیلتر**: منوی کشویی انتخاب رمزارز، ورودی جستجو، انتخابگر تاریخ (Flatpickr)
  3. **محل بارگذاری اخبار**: <div id="newsGrid">
  4. **محل بارگذاری قیمت لحظه‌ای**: <div id="priceDisplay">
  5. **لودر انیمیشنی**: سه آیکون رمزارز با loader.css
  6. **تم تاریک/روشن**: سوئیچ با ذخیره در localStorage
* **templates/post.html**  
  صفحهٔ جزئیات خبر:
  1. عنوان، تاریخ و منبع در هدر
  2. متن کامل خبر در بخش بدنه با کلاس‌های Tailwind و Prose برای خوانایی
  3. لینک بازگشت به صفحه اصلی

**استایل‌ها و CSS**

* **Tailwind CSS**
  + استفاده از کلاس‌های Utility-first برای سریع‌تر و خواناتر نوشتن استایل
* **static/css/styles.css**
  + استایل‌های اختصاصی پروژه (فونت، اندازه‌ها، فاصله‌ها(
* **static/css/loader.css**
  + تعریف انیمیشن چرخش آیکون‌ها با @keyframes rotate
  + تأخیرهای متفاوت برای هر آیکون

**اسکریپت‌های JavaScript**

* **بارگذاری اولیه**
  1. فراخوانی /api/coins برای پر کردن منوی فیلتر
  2. فراخوانی هم‌زمان /api/news و /api/prices
* **بارگذاری اخبار (loadNews)**

function loadNews() {

// مدیریت حالت بارگذاری و پارامترها

fetch(`/api/news?${params.toString()}`)

.then(res => res.json())

.then(displayNews)

.finally(() => { isLoading = false; });

}

* 1. استفاده از offset و limit برای صفحه‌بندی بی‌پایان
  2. **Cache محلی**: با newsCache از درخواست‌های تکراری جلوگیری می‌شود
* **نمایش اخبار (displayNews)**
  1. کلون کردن template
  2. پرکردن عنوان، تاریخ/زمان و پیش‌نمایش متن
  3. تولید برچسب‌های رمزارز با price-tooltip برای نمایش قیمت لحظه‌ای و تاریخی
  4. اضافه کردن رویداد کلیک برای باز کردن کادر (Modal)
* **لودر و مدیریت حالت‌ها**
  1. المان <div id="loading"> هنگام بارگذاری نمایش داده می‌شود
  2. پس از بارگذاری یا خطا، مخفی می‌شود

**واکنش‌گرایی (Responsive Design)**

* **Flex و Grid**
  + استفاده از کلاس‌های Tailwind مانند flex, space-y-4, max-w-4xl
* **نقطه شکست (Breakpoints)**
  + تنظیم w-80 برای سایدبار در دسکتاپ
  + در موبایل، سایدبار با جاوااسکریپت مخفی/نمایش می‌شود
* **اندازه فونت و فاصله**
  + text-sm, text-lg, p-4 و … برای خوانایی در نمایشگرهای مختلف

**تعاملات کاربر**

* **فیلتر بر اساس رمزارز**
  + انتخاب از منوی کشویی و بارگذاری مجدد اخبار
* **جستجوی متنی**
  + فیلتر در عنوان خبر
* **انتخاب بازهٔ زمانی**
  + استفاده از Flatpickr با گزینهٔ enableTime
* **تم تاریک/روشن**
  + ذخیره حالت در localStorage و اعمال کلاس روی document.documentElement
* **کادر (Modal)**
  + باز و بسته کردن داینامیک با افزودن/حذف کلاس‌های active و hidden
  + جلوگیری از بسته‌شدن هنگام کلیک درون محتوا و بستن تنها با کلیک بیرون

**مدیریت کش و بهینه‌سازی عملکرد**

**کش قیمت‌ها در حافظه**

* **هدف:** کاهش تعداد درخواست‌ها به API خارجی (CoinGecko) و صرفه‌جویی در زمان و پهنای‌باند.
* **مکانیزم:**
  1. **price\_cache**: یک دیکشنری سراسری در اسکریپت main.py نگهداری می‌شود.
  2. **قفل (Lock):** با استفاده از ماژول threading.Lock، دسترسی به price\_cache در تردهای هم‌زمان ایمن می‌شود.
  3. **اعتبار ۵ دقیقه‌ای:**
     + هر دادهٔ کش‌شده همراه با timestamp به‌روز می‌شود.
     + اگر از آخرین به‌روزرسانی کمتر از ۳۰۰ ثانیه (۵ دقیقه) گذشته باشد، دادهٔ کش‌شده استفاده می‌شود.
  4. **دکوریتور @lru\_cache(maxsize=100)** روی تابع get\_cached\_price اضافه شده تا کنترل کش در سطح تابع نیز برقرار باشد.

**ایجاد ایندکس‌ها در PostgreSQL**

* **هدف:** تسریع عملیات جستجو و بازیابی داده‌ها در جداول بزرگ.
* **ایندکس‌های پیاده‌شده:**
  1. **idx\_crypto\_news\_published\_date**
     + روی ستون published\_date در جدول crypto\_news
     + جهت مرتب‌سازی نزولی اخبار بر اساس تاریخ انتشار
  2. **idx\_crypto\_news\_title**
     + GIN index با to\_tsvector('english', title)
     + برای جستجوی سریع متنی در عنوان
  3. **idx\_news\_coins\_news\_id** و **idx\_news\_coins\_coin\_id**
     + برای سرعت‌بخشیدن به JOIN بین crypto\_news و news\_coins
  4. **idx\_crypto\_prices\_coin\_timestamp**
     + ترکیبی روی (coin\_id, timestamp DESC)
     + جهت واکشی سریع آخرین قیمت هر رمزارز
* **تغییرات:**  
  با اجرای تابع recreate\_indexes() در ابتدای برنامه، ایندکس‌ها در صورت وجود حذف و مجدداً ایجاد می‌شوند تا ساختار بهینه حفظ شود.

**جلوگیری از اخبار تکراری**

* **قید یکتا (UNIQUE)**
  + در ستون url جدول crypto\_news
  + از درج رکورد با URL مشابه جلوگیری می‌کند.
* **بررسی پیش از درج**
  + متد store\_news() ابتدا با یک SELECT بررسی می‌کند که آیا خبر با آن URL قبلاً ذخیره شده است یا نه.
  + در صورت وجود:
    - تنها در صورت انتشار مجدد با تاریخ جدیدتر، رکورد با UPDATE به‌روزرسانی می‌شود.
  + این کار از ورود داده‌های تکراری یا ترتیب نادرست رکوردها جلوگیری می‌کند.

**بهبود زمان پاسخ API**

* **صفحه‌بندی (Pagination)**
  + پارامترهای limit و offset برای API /api/news فراهم شده تا در هر درخواست تنها تعداد محدودی رکورد خوانده شود.
* **استفاده از CTE و JOIN بهینه**
  + در کوئری دریافت اخبار، با استفاده از CTE (filtered\_news) ابتدا فیلتر روی crypto\_news انجام می‌شود، سپس JOIN با news\_coins و آخرین قیمت از crypto\_prices صورت می‌پذیرد.
* **فشرده‌سازی پاسخ**
  + Flask می‌تواند با فعال کردن Flask-Compress خروجی JSON و HTML را GZIP کند (می‌تواند به‌عنوان بهینه‌سازی آتی اضافه شود).
* **کش HTTP (Future Improvement)**
  + هدرهای HTTP Cache-Control می‌تواند برای منابع ایستا (static/) تنظیم شود تا مرورگر کاربر بارگذاری مجدد نداشته باشد.

**ملاحظات امنیتی**

**نگهداری ایمن کلیدهای API**

* **محل ذخیره:**  
  تمامی کلیدهای حساس ( COINDESK\_API\_KEY, MESSARI\_API\_KEYو اطلاعات دیتابیس) در فایل .env ذخیره می‌شوند و این فایل در .gitignore قرار دارد.
* **بارگذاری پویا:**  
  با استفاده از کتابخانهٔ python-dotenv، متغیرهای محیطی تنها در زمان اجرا بارگذاری می‌شوند و در کد صریح نوشته نمی‌شوند.
* **حداقل سطح دسترسی:**  
  کلیدها ایجادشده در سرویس‌های خارجی باید سطح دسترسی لازم (فقط خواندن داده) را داشته باشند تا در صورت نشت، آسیب به حداقل برسد.

**اعتبارسنجی و پاک‌سازی ورودی‌ها**

* **پارامترهای ورودی API**
  + تمامی پارامترهای کوئری (coin, search, date\_from, date\_to, limit, offset) قبل از استفاده در کوئری SQL بررسی و به‌صورت پارامترایز شده (%s) به execute منتقل می‌شوند تا از SQL Injection جلوگیری شود.
* **فیلتر تاریخ**
  + ورودی‌های date\_from و date\_to از فرمت YYYY-MM-DD یا YYYY-MM-DD HH:MM پشتیبانی کرده و در صورت نامعتبر بودن، به صورت پیش‌فرض نادیده گرفته می‌شوند.
* **پاک‌سازی متن**
  + در ذخیره‌سازی عنوان و محتوا، هرگونه نویسهٔ غیرمجاز با .encode('utf-8', 'ignore').decode('utf-8') فیلتر می‌شود تا از ثبت کاراکترهای مخرب جلوگیری شود.

**محدود کردن نرخ درخواست‌ها**

* **درخواست به APIهای خارجی**
  + **CoinDesk**: در صورت دریافت خطای محدودیت نرخ (Rate Limit)، درخواست مجدد پس از مدت کوتاهی برنامه‌ریزی می‌شود (می‌توان Retry با Backoff افزایشی اضافه کرد).
  + **CoinGecko** و **Messari**: با استفاده از کش ۵ دقیقه‌ای و LRU Cache تعداد درخواست‌ها به حداقل می‌رسد.
* **درخواست‌های کاربر به سرویس**
  + می‌توان از افزونه‌هایی مانند Flask-Limiter برای محدودسازی تعداد درخواست‌ها در بازهٔ زمانی مشخص استفاده کرد تا حملات DDoS سطح پایین کنترل شود.

**تنظیم CORS و HTTPS**

* **CORS (Cross-Origin Resource Sharing)**
  + در نسخهٔ فعلی تنها از دامنهٔ خود برنامه (localhost:5000 یا دامنهٔ استقرار) درخواست پذیرفته می‌شود.
  + برای توسعه می‌توان از Flask-CORS استفاده کرده و لیست مجاز منابع را به‌صورت دقیق تعریف نمود.
* **HTTPS**
  + هنگام استقرار در محیط تولید، باید TLS/SSL فعال شود تا تمام ارتباطات رمزنگاری گردد.
  + می‌توان از پروکسی معکوس (Reverse Proxy) مانند Nginx یا سرویس‌های مدیریت‌شده‌ی SSL (Cloudflare, Let’s Encrypt) استفاده نمود.

**آزمون و تضمین کیفیت**

**تست‌های واحد (Unit Tests)**

* **هدف:** اعتبارسنجی جداگانهٔ کوچک‌ترین واحدهای کد (توابع و متدها)
* **ابزار پیشنهادی:** pytest
* **محدودهٔ تست:**
  + check\_coin\_mentions() با نمونه‌های مختلف متن برای اطمینان از تشخیص درست نام و نماد رمزارز
  + get\_cached\_price() و update\_price\_cache() برای تضمین عملکرد صحیح کش
  + توابع کمکی مثل get\_db\_connection() برای بررسی اتصال بدون خطا (با Mocking پارامترها)
* **نمونه کد:**

def test\_check\_coin\_mentions\_simple():

article = {"title": "Bitcoin and ETH rally", "body": ""}

assert check\_coin\_mentions(article) == {"bitcoin", "ethereum"}

**تست‌های یکپارچه‌سازی (Integration Tests)**

* **هدف:** اعتبارسنجی تعامل میان اجزای مختلف (بک‌اند و پایگاه داده)
* **ابزار پیشنهادی:**
  + pytest به‌همراه پلاگین pytest-postgresql یا تست در برابر یک دیتابیس تست محلی
  + requests برای ارسال HTTP به سرور Flask
* **محدودهٔ تست:**
  + فراخوانی /api/news پس از درج چند خبر تستی و اطمینان از ساختار JSON صحیح
  + درج اخبار و بررسی ذخیره‌شدن رابطه‌ها در جداول crypto\_news، crypto\_prices و news\_coins
  + تست صفحه‌بندی با پارامترهای limit و offset

**تست دستی فرانت‌اند**

* **هدف:** اطمینان از عملکرد صحیح رابط کاربری در مرورگرهای مختلف
* **مراحل:**
  1. تست بارگذاری اولیهٔ صفحه و نمایش لودینگ انیمیشن
  2. اعمال فیلتر رمزارز، جستجو و بازهٔ زمانی و اطمینان از بازخوانی صحیح کارت‌ها
  3. تست باز/بسته شدن Modal جزئیات خبر
  4. تست تم تاریک/روشن و ماندگاری حالت در localStorage
* **ابزار کمکی:** Chrome DevTools و Lighthouse برای اندازه‌گیری عملکرد

**معیارهای پذیرش**

1. **پاسخ صحیح API**
   * همهٔ نقاط پایانی با پارامترهای معتبر و نامعتبر به‌درستی پاسخ دهند (کد وضعیت و فرمت خروجی)
2. **عدم خطا در لاگ‌ها**
   * هنگام اجرای tracker.run() یا درخواست‌های API خطای غیرمنتظره ثبت نشود
3. **کارایی قابل قبول**
   * بار اولیهٔ صفحه زیر ۲ ثانیه
   * پاسخ API /api/news برای ۲۰ نتیجه زیر ۵۰۰ میلی‌ثانیه در دیتابیس کم‌بار
4. **امنیت پایه**
   * تست نفوذ ساده برای SQL Injection، XSS در فیلدهای جستجو
5. **پایداری کش**
   * تست اعتبارسنجی کش قیمت با شبیه‌سازی تاخیر در API خارجی

**راهنمای راه‌اندازی و استقرار**

**نصب پیش‌نیازها**

1. **سیستم‌عامل**
   * لینوکس (اوبونتو 20.04+) یا macOS
   * ویندوز (ترجیحاً با WSL2)
2. **نصب Python**
   * نسخهٔ ۳.۸ یا بالاتر
   * بررسی نسخه:

python --version

1. **نصب PostgreSQL**
   * همراه با ابزار psql
   * در اوبونتو:

sudo apt update

sudo apt install postgresql postgresql-contrib

1. **نصب Git**

sudo apt install git

**پیکربندی محیط (Virtualenv & .env)**

1. **کلون مخزن**

git clone https://github.com/Erm1a/crypto-news-tracker.git

cd crypto-news-tracker

1. **ایجاد و فعال‌سازی محیط مجازی**

python -m venv .venv

source .venv/bin/activate # macOS/Linux

# .venv\Scripts\activate # Windows (PowerShell)

1. **نصب وابستگی‌ها**

pip install --upgrade pip

pip install -r requirements.txt

1. **ساخت فایل .env**  
   در شاخهٔ ریشهٔ پروژه، فایلی با نام .env بسازید و مقادیر زیر را تنظیم کنید:

DB\_NAME=Crypto\_News

DB\_USER=your\_db\_user

DB\_PASSWORD=your\_db\_password

DB\_HOST=localhost

DB\_PORT=5432

COINDESK\_API\_KEY=your\_coindesk\_api\_key

MESSARI\_API\_KEY=your\_messari\_api\_key # اختیاری

**اجرای محلی**

1. **ایجاد جداول و ایندکس‌ها**

python main.py # با هر اجرا، ابتدا create\_tables و recreate\_indexes اجرا می‌شود.

1. **اجرا و توسعه**  
   پروژه با فعال بودن ترد جمع‌آوری اخبار و سرور Flask هم‌زمان اجرا می‌شود.
   * آدرس: http://localhost:5000/
   * مود debug فعال است؛ لاگ‌ها در ترمینال نمایش داده می‌شوند.
2. **توقف اجرا**
   * با زدن Ctrl+C در ترمینال سرور متوقف می‌شود.
   * ترد جمع‌آوری اخبار نیز خاتمه می‌یابد.

**استقرار روی سرور Docker:**

**الف) با استفاده از Gunicorn و Nginx (لینوکس)**

1. **نصب Gunicorn**

pip install gunicorn

1. **راه‌اندازی Gunicorn**

cd crypto-news-tracker

source .venv/bin/activate

gunicorn --bind 0.0.0.0:8000 main:app --workers 4

1. **تنظیم Nginx**
   * در فایل پیکربندی /etc/nginx/sites-available/crypto\_news:

server {

listen 80;

server\_name your\_domain.com;

location / {

proxy\_pass http://127.0.0.1:8000;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

location /static/ {

alias /path/to/crypto-news-tracker/static/;

}

}

* + فعال‌سازی و ری‌استارت Nginx:

ln -s /etc/nginx/sites-available/crypto\_news /etc/nginx/sites-enabled/

sudo systemctl restart nginx

1. **فعال‌سازی HTTPS**
   * با Certbot و Let’s Encrypt:

sudo apt install certbot python3-certbot-nginx

sudo certbot --nginx -d your\_domain.com

**ب) با استفاده از Docker**

1. **نوشتن Dockerfile**

FROM python:3.10-slim

WORKDIR /app

COPY . .

RUN pip install --upgrade pip && \

pip install -r requirements.txt

# متغیرهای محیطی از طریق docker-compose یا -e

CMD ["gunicorn", "--bind", "0.0.0.0:5000", "main:app", "--workers", "2"]

1. **نوشتن docker-compose.yml**

version: '3.8'

services:

db:

image: postgres:14

environment:

POSTGRES\_DB: Crypto\_News

POSTGRES\_USER: your\_db\_user

POSTGRES\_PASSWORD: your\_db\_password

volumes:

- db\_data:/var/lib/postgresql/data

web:

build: .

ports:

- "5000:5000"

environment:

DB\_NAME: Crypto\_News

DB\_USER: your\_db\_user

DB\_PASSWORD: your\_db\_password

DB\_HOST: db

DB\_PORT: 5432

COINDESK\_API\_KEY: your\_coindesk\_api\_key

MESSARI\_API\_KEY: your\_messari\_api\_key

depends\_on:

- db

volumes:

db\_data:

1. **راه‌اندازی با Docker Compose**

docker-compose up -d

1. **دسترسی**
   * برنامه روی http://localhost:5000 یا آدرس سرور قابل دسترسی است.

**مدیریت عملیات و نگهداری**

**نظارت بر عملکرد (Logging & Monitoring)**

* **Logging**
  + استفاده از ماژول logging پایتون به‌جای print برای ثبت لاگ‌های سطوح مختلف (DEBUG, INFO, WARNING, ERROR).
  + هدایت لاگ‌ها به فایل یا سرویس‌هایی مانند ELK Stack (Elasticsearch, Logstash, Kibana) یا Graylog.
* **Monitoring**
  + ابزارهایی مانند Prometheus و Grafana برای پایش هزاران متریک:
    - نرخ درخواست‌های API
    - زمان پاسخ‌دهی
    - مصرف حافظه و CPU
  + تعریف آلارم (Alert) در صورت افزایش خطاهای 5xx یا تاخیر زیاد در پاسخ‌ها.

**بک‌آپ و بازیابی پایگاه داده**

* **بک‌آپ خودکار**
  + پیکربندی pg\_dump دوره‌ای (روزانه یا ساعتی براساس حجم داده) برای صادرکردن دیتابیس
  + ذخیرهٔ فایل‌های بک‌آپ در مکان امن (S3، سرور جداگانه)
* **بازیابی (Restore)**
  + استفاده از pg\_restore یا واردکردن اسکریپت SQL بک‌آپ
  + تست دوره‌ای فرآیند بازیابی برای اطمینان از صحت داده

**به‌روزرسانی و نگهداری دوره‌ای**

* **به‌روزرسانی وابستگی‌ها**
  + بررسی دوره‌ای نسخه‌های جدید بسته‌ها (pip list --outdated)
  + اعمال پچ‌های امنیتی و رفع باگ‌ها در نسخه‌های جدید
* **مهاجرت‌های پایگاه داده**
  + در صورت تغییر ساختار جداول یا افزودن ستون/ایندکس جدید، استفاده از ابزارهایی مانند Alembic برای مدیریت مهاجرت
* **نظافت کش و فایل‌های قدیمی**
  + حذف رکوردهای crypto\_prices قدیمی (بیش از یک سال) در یک اسکریپت زمان‌بندی‌شده
  + پاک‌سازی لاگ‌های قدیمی با نگه‌داری محدود (مثلاً ۳۰ روز)

**مستندسازی عملیات**

* **Runbook**
  + ایجاد مستنداتی برای مراحل معمولی:
    - راه‌اندازی محیط
    - بررسی لاگ‌ها
    - بازیابی بک‌آپ
  + قرارگیری Runbook در مخزن یا سیستم مدیریت دانش (Confluence)
* **On-call Handbook**
  + فهرست تماس‌ها و مسئولیت‌ها
  + چک‌لیست‌های سریع برای مواقع قطعی (Downtime)

**نقاط قابل بهبود و توسعه‌های آینده**

**اضافه کردن منابع خبری بیشتر**

* **یکپارچه‌سازی با APIهای جدید**
  + افزونۀ پشتیبانی از منابع مطرح مانند CryptoSlate، The Block، CoinTelegraph
  + پیاده‌سازی پارسرهای سفارشی برای منابع فاقد API رسمی
* **RSS Feed**
  + استفاده از فیدهای RSS برای دریافت سریع‌تر اخبار
  + ترکیب بین دادهٔ RSS و API برای پوشش گسترده‌تر

**پشتیبانی از رابط کاربری پیشرفته**

* **فریم‌ورک‌های مدرن فرانت‌اند**
  + مهاجرت به React یا Vue.js برای ساخت کامپوننت‌های قابل بازیافت
  + مدیریت وضعیت با Redux یا Vuex
* **داشبورد تحلیلی**
  + نمایش نمودارهای قیمت و حجم خبر
  + تحلیل احساسات (Sentiment Analysis) با نمودارهای زمان‌مند
* **اعلان و نوتیفیکیشن**
  + ارسال ایمیل یا پیام‌های Push برای اخبار مهم یا نوسان‌های بزرگ قیمتی
  + تنظیم Threshold توسط کاربر

**یادگیری ماشین برای تحلیل احساسات**

* **تحلیل احساسات متن خبر**
  + استفاده از مدل‌های NLP (مثل BERT فارسی/انگلیسی) برای تعیین مثبت، منفی یا خنثی بودن هر مقاله
  + ذخیره و نمایش امتیاز احساسات کنار هر خبر
* **پیش‌بینی تأثیر قیمت**
  + مدلسازی رابطه بین احساسات اخبار و نوسان قیمت
  + ارائهٔ پیش‌بینی کوتاه‌مدت یا میان‌مدت قیمت رمزارز

**گسترش پایگاه داده و گزارش‌گیری**

* **افزودن شاخص‌های بازار**
  + ذخیرهٔ داده‌های کلی بازار (Market Cap Total، حجم معاملات کلی)
  + جداول جداگانه برای نمودارهای تاریخی
* **گزارش‌ساز سفارشی**
  + امکان استخراج گزارش PDF/Excel از داده‌های خبر و قیمت
  + تعریف فیلترهای پیچیده برای ساخت گزارش‌های هوش تجاری

**بهینه‌سازی مقیاس و خوشه‌بندی**

* **استفاده از کش خارجی (Redis/Memcached)**
  + جایگزینی کش داخلی Python با Redis برای مقیاس‌پذیری توزیع‌شده
* **خوشه‌بندی PostgreSQL**
  + Replica برای خواندن سنگین
  + Partitioning جداول بزرگ (crypto\_news تاریخی)

**مستندسازی و نمونه‌های برنامه‌نویسی**

* **ایجاد SDK**
  + توسعه کتابخانۀ پایتون یا جاوااسکریپت برای استفاده از API پروژه
* **مستندات تعاملی**
  + استفاده از Swagger/OpenAPI برای مستندسازی دقیق REST API
  + صفحهٔ API Explorer برای آزمایش مستقیم نقاط پایانی

**نتیجه‌گیری**

در این مستند، پروژه «Crypto News Tracker» از ابتدا تا انتها بررسی شد. این سامانه با بهره‌گیری از معماری سه‌لایه (فرانت‌اند، بک‌اند و پایگاه داده) و سرویس‌های خارجی معتبر، توانایی گردآوری اخبار کریپتوکارنسی، استخراج نام رمزارزها، ذخیره و نگاشت دقیق اطلاعات قیمتی لحظه‌ای و تاریخی را دارد. نکات کلیدی عبارت‌اند از:

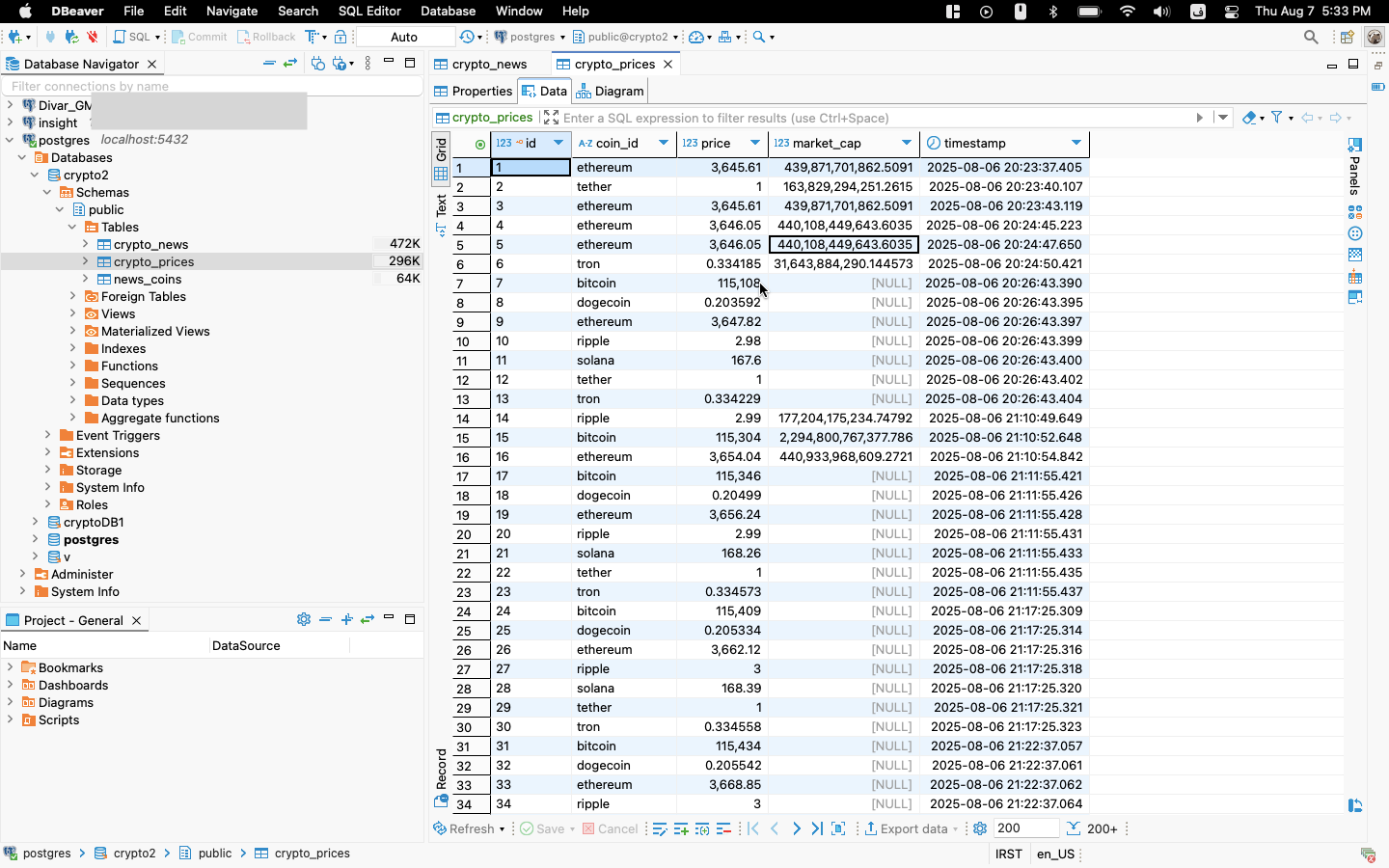
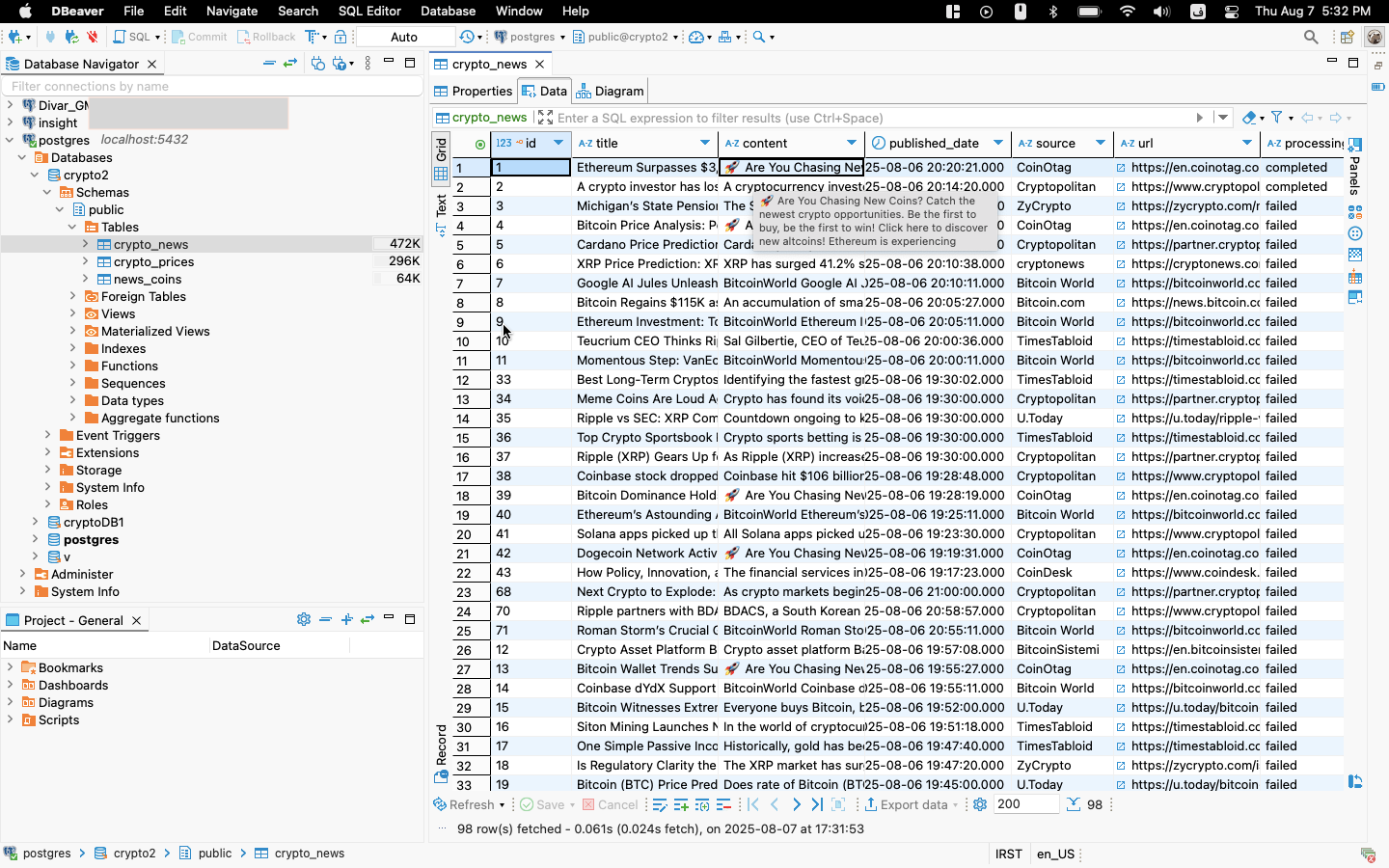
* **یکپارچگی منابع:** جمع‌آوری اخبار از CoinDesk و امکان افزودن منابع بیشتر در آینده.
* **مدیریت داده‌ها:** ذخیره‌سازی بهینه با استفاده از PostgreSQL، ایندکس‌گذاری و جلوگیری از داده‌های تکراری.
* **بهینه‌سازی عملکرد:** کش داخلی، صفحه‌بندی API و طراحی کوئری‌های کارآمد.
* **امنیت و پایداری:** نگهداری ایمن کلیدهای API، پارامترایز کردن کوئری‌ها، تست‌های متعدد و قابلیت استقرار در محیط‌های مختلف.
* **قابلیت توسعه:** طراحی ماژولار پروژه اجازهٔ افزودن قابلیت‌هایی مانند تحلیل احساسات، داشبورد تحلیلی و مقیاس‌پذیری بالا را می‌دهد.

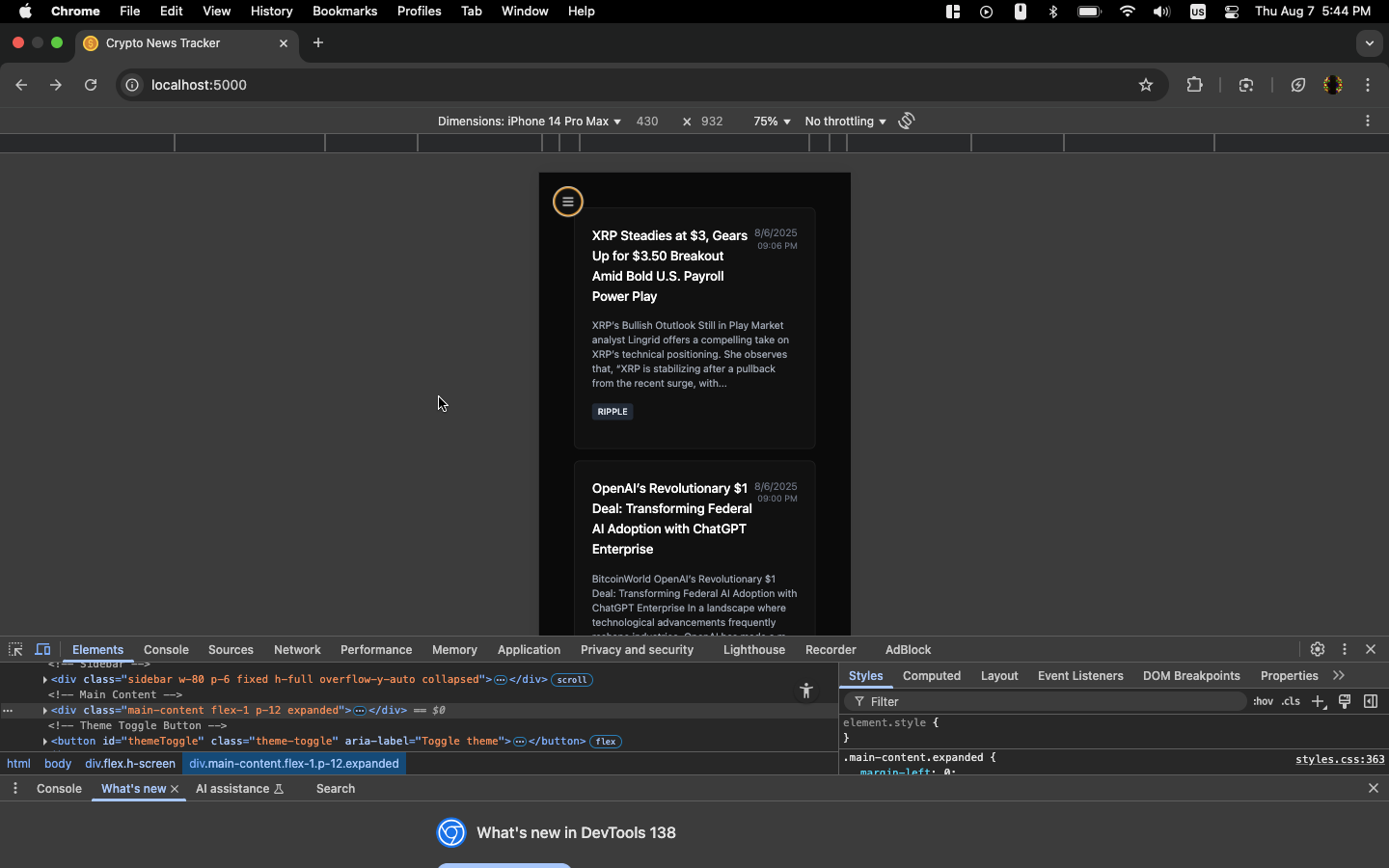
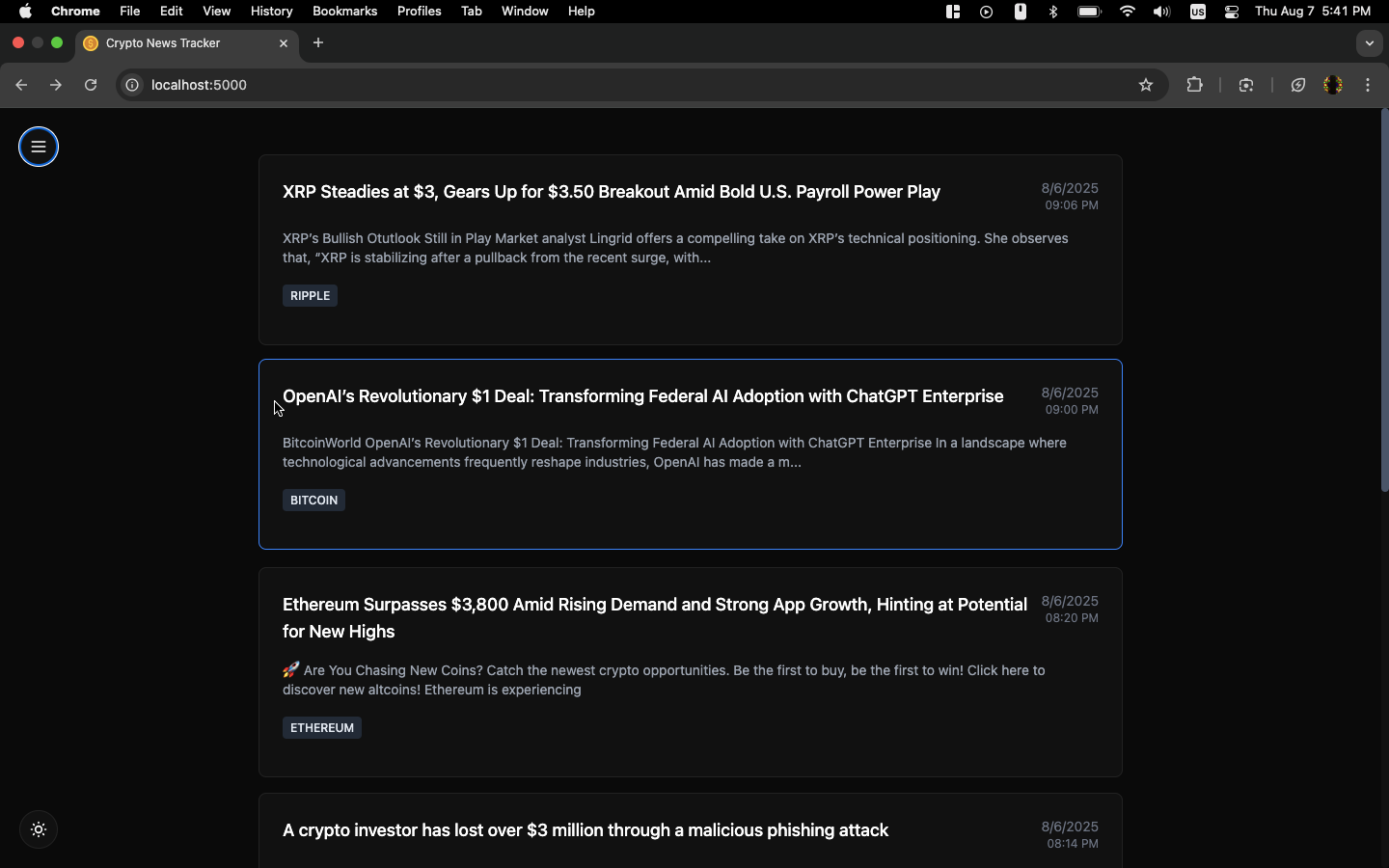
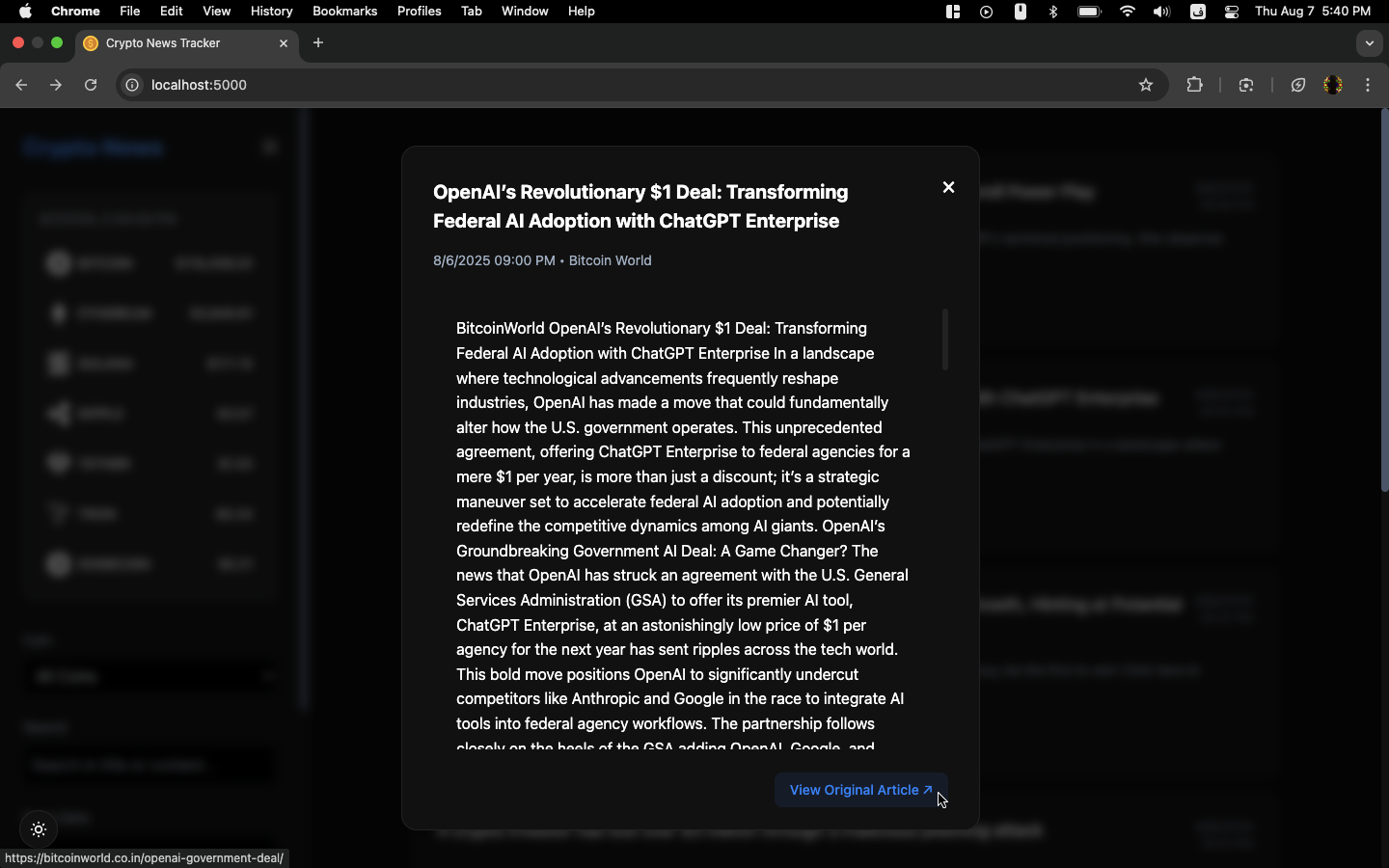
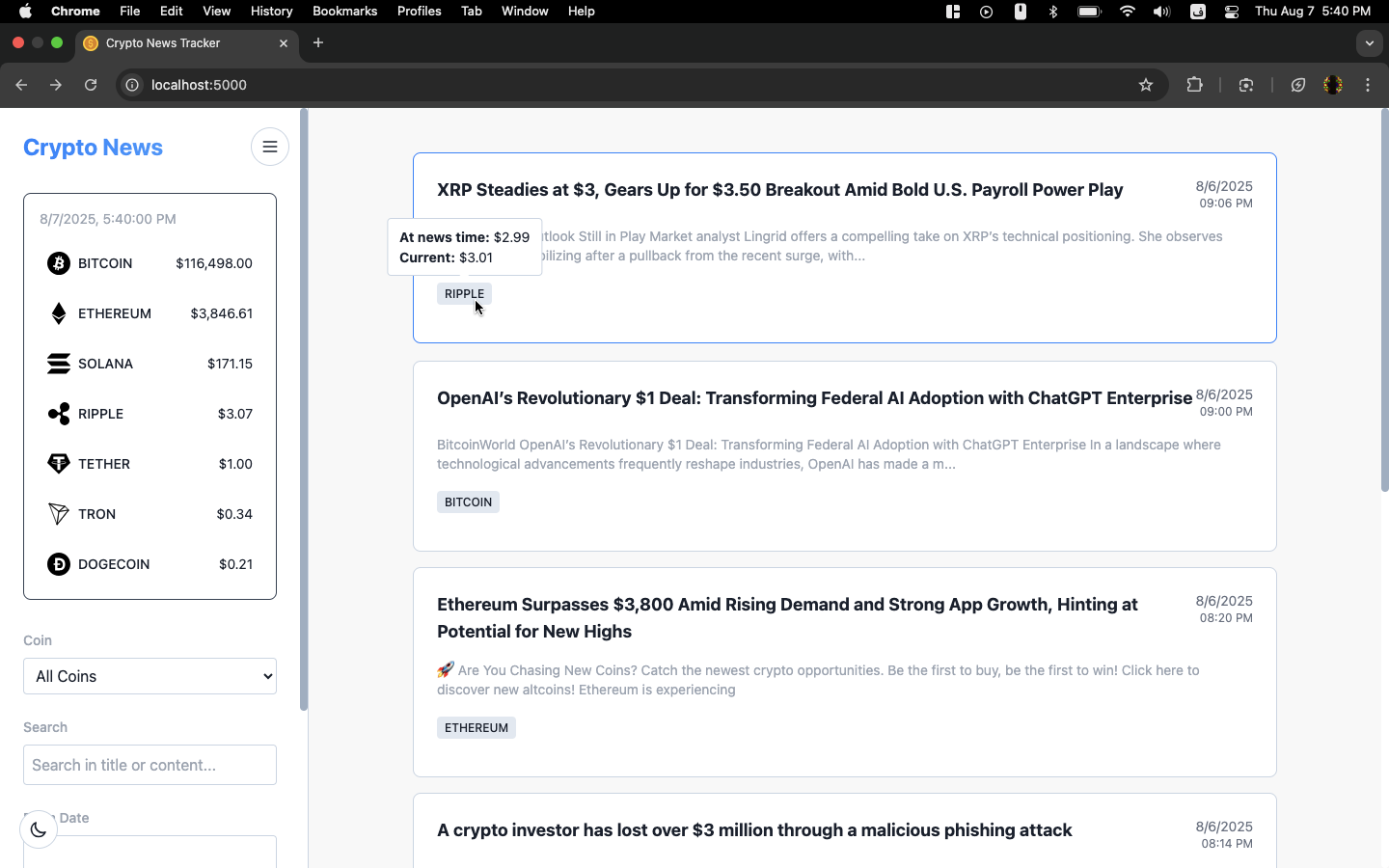
با توجه به پوشش کامل نیازهای اولیه، این پروژه می‌تواند به‌عنوان پایه‌ای قوی برای سامانه‌های پیچیده‌تر خبری و تحلیلی در حوزهٔ کریپتوکارنسی به کار رود.

**ضمائم**

**15.1 فهرست واژگان**

**ساختار کامل بانک اطلاعاتی (ER Diagram) **

**نمایی از جداول دیتابیس به همراه دیتا در برنامه DBeaver**   


**نمایی از وبسایت به صورت ران لوکال**

**منابع و مآخذ**

1. مستندات رسمی فلسک Flask: [https://flask.palletsprojects.com](https://flask.palletsprojects.com/)
2. راهنمای دیتابیس پستگرس PostgreSQL: <https://www.postgresql.org/docs>
3. مستندات کوین گکو CoinGecko API: <https://www.coingecko.com/en/api>
4. مستندات کوین دسک CoinDesk API (Data API): <https://www.coindesk.com/coindesk-api>
5. مستندات میزوری : <https://messari.io/api> Messari API
6. لینک گیت هاب پروژه GitHub: <https://github.com/Erm1a/crypto-news-tracker>