Həftə 4 — Async Programming, Integration & Deployment: Ta

Gün 1 — Introduction to Asynchronous Programming

Mövzular: asyncio module, async/await, defining and running async tasks

---

Assignment 1.1 — Asinxron Veb Scraper

Bir neçə URL-dən eyni vaxtda data toplayan asinxron scraper yazın.

Initial Data:

```
urls_to_scrape = [
    {"name": "JSONPlaceholder Posts", "url": "https://jsonplaceholder.typicode.com/posts", "type": "api"},
    {"name": "JSONPlaceholder Users", "url": "https://jsonplaceholder.typicode.com/users", "type": "api"},
    {"name": "JSONPlaceholder Comments", "url": "https://jsonplaceholder.typicode.com/comments?postId=1", "type": "api"},
    {"name": "HTTPBin Get", "url": "https://httpbin.org/get", "type": "api"},
    {"name": "HTTPBin IP", "url": "https://httpbin.org/ip", "type": "api"},
    {"name": "HTTPBin Headers", "url": "https://httpbin.org/headers", "type": "api"},
    {"name": "HTTPBin Delay 1s", "url": "https://httpbin.org/delay/1", "type": "slow"},
    {"name": "HTTPBin Delay 2s", "url": "https://httpbin.org/delay/2", "type": "slow"},
    {"name": "HTTPBin UUID", "url": "https://httpbin.org/uuid", "type": "api"},
    {"name": "GitHub API Zen", "url": "https://api.github.com/zen", "type": "api"},
]
```

Tələblər:

1. `aiohttp` ilə `async def fetch_url(session, url_info) -> dict` yazın — URL-dən data alıb status, response time, body size qaytarsın
2. `asyncio.gather()` ilə bütün URL-ləri paralel çəkin
3. `asyncio.Semaphore(3)` ilə eyni vaxtda maksimum 3 sorğu göndərin
4. `asyncio.wait_for(coro, timeout=5)` ilə timeout tətbiq edin
5. **Sinxron müqayisə**: eyni URL-ləri `requests` ilə ardıcıl çəkin, vaxt fərqini ölçün
6. Nəticəni JSON faylına yazın, gözəl hesabat çap edin:

```
URL                      Status  Vaxt (ms)  Ölçü (KB)

JSONPlaceholder Posts       200       145      82.3
HTTPBin Delay 2s            200      2120       0.5
...

Sinxron: 12.5 san | Asinxron: 2.8 san | Sürət: 4.5x
```

Assignment 1.2 — Asinxron Tapşırıq Növbəsi (Task Queue)

Müxtəlif növ tapşırıqları asinxron idarə edən növbə sistemi yazın.

Tələblər:

1. `asyncio.Queue` ilə tapşırıq növbəsi yaradın
2. Tapşırıq tipləri (hər biri `async def`):
   - `email_task` — e-poçt göndərməni simulyasiya (0.5-2 san gecikdirmə)
   - `report_task` — hesabat hazırlamağı simulyasiya (1-3 san)
   - `notification_task` — bildiriş göndərməni simulyasiya (0.2-0.8 san)
   - `backup_task` — backup almağı simulyasiya (2-5 san)
1. **Producer**: tapşırıqları növbəyə əlavə etsin (fərqli prioritetlərlə)
2. **Consumer** (3 ədəd): növbədən paralel oxuyub icra etsin — `asyncio.create_task` ilə
3. Hər tapşırığın statusunu izləyin: `pending → running → completed/failed`
4. `asyncio.wait` ilə bütün tapşırıqların bitməsini gözləyin

Initial Data:

```
task_batch = [
    {"type": "email", "priority": 2, "data": {"to": "eli@mail.az", "subject": "Hesabat", "body": "Aylıq hesabat hazırdır"}},
    {"type": "report", "priority": 1, "data": {"name": "Aylıq Satış", "format": "pdf", "period": "2026-03"}},
    {"type": "notification", "priority": 3, "data": {"user": "leyla", "message": "Yeni tapşırıq təyin edildi"}},
    {"type": "backup", "priority": 1, "data": {"source": "/data/db", "destination": "/backup/db_20260317"}},
    {"type": "email", "priority": 2, "data": {"to": "tural@mail.az", "subject": "Görüş", "body": "Sabah saat 10:00"}},
    {"type": "notification", "priority": 3, "data": {"user": "nigar", "message": "Deadline yaxınlaşır"}},
    {"type": "report", "priority": 2, "data": {"name": "Həftəlik KPI", "format": "csv", "period": "2026-W11"}},
    {"type": "backup", "priority": 1, "data": {"source": "/data/logs", "destination": "/backup/logs_20260317"}},
    {"type": "email", "priority": 3, "data": {"to": "rashad@mail.az", "subject": "Yenilik", "body": "Sistem yeniləndi"}},
    {"type": "notification", "priority": 2, "data": {"user": "sabina", "message": "Review tələb olunur"}},
]
```

Assignment 1.3 — Asinxron Chat Server Simulyatoru

Bir neçə "istifadəçi" arasında asinxron mesajlaşma sistemini simulyasiya edin.

Tələblər:

1. `ChatRoom` sinfi: `asyncio.Queue` ilə mesaj növbəsi, istifadəçi siyahısı
2. `User` sinfi: ad, `async def send_message(room, text)`, `async def listen(room)`
3. Hər istifadəçi ayrı `asyncio.Task` olaraq işləsin:
   - Təsadüfi intervallarla mesaj göndərsin (0.5-3 san)
   - Digər istifadəçilərin mesajlarını alsın
1. `async def broadcast(room, message, sender)` — mesajı bütün istifadəçilərə yayın
2. `asyncio.Event` ilə otağı bağlama/açma funksionallığı
3. 20 saniyə sonra simulyasiyanı dayandırın, statistika çap edin: kim neçə mesaj göndərdi, ən aktiv istifadəçi, mesaj tezliyi

Initial Data:

```
users_config = [
    {"name": "Əli", "typing_speed": 1.5, "messages": [
        "Salam hamıya!", "Layihə necə gedir?", "Mən backend hissəni bitirdim",
        "Review lazımdır", "Sabah deploy edərik"]},
    {"name": "Leyla", "typing_speed": 1.0, "messages": [
        "Salam!", "Frontend hazırdır", "Testləri yazdım",
        "API endpointlər işləyir?", "Əla iş!"]},
    {"name": "Tural", "typing_speed": 2.0, "messages": [
        "Mən buradayam", "Database migration lazımdır", "Bug tapdım",
        "Fix göndərdim", "Merge edək"]},
    {"name": "Nigar", "typing_speed": 0.8, "messages": [
        "QA testləri başladım", "3 bug tapdım", "Report hazırdır",
        "Regression keçdi", "Deploy-a icazə verirəm"]},
]
```

Bonus Assignment 1.B — Refaktor: Futures və Exception Handling

> *Gün 2-nin mövzularını öyrəndikdən sonra edin.*

Assignment 1.1 (Veb Scraper) kodunu refaktor edin:

1. `concurrent.futures.ThreadPoolExecutor` ilə alternativ sinxron versiya yazın, `asyncio` versiyası ilə müqayisə edin
2. Hər URL çağırışı üçün `asyncio.ensure_future` + callback pattern istifadə edin
3. Xəta hallarını `asyncio.gather(return_exceptions=True)` ilə idarə edin
4. Retry məntiqi əlavə edin: uğursuz URL-ləri exponential backoff ilə 3 dəfə təkrarlayın

---

Gün 2 — Advanced Asynchronous Techniques

Mövzular: futures, coroutines, async exception handling, concurrent.futures

---

Assignment 2.1 — Distributor Emal Sistemi

CPU-intensive və I/O-intensive tapşırıqları optimal strategiya ilə bölüşdürən hibrid sistem yazın.

Tələblər:

1. **I/O-bound tapşırıqlar** (asyncio + aiohttp):
   - 10 API-dən paralel data çəkmə
   - Fayl oxuma/yazma simulyasiyası

1. **CPU-bound tapşırıqlar** (concurrent.futures.ProcessPoolExecutor):
   - Böyük ədədlərin faktorializasiyası
   - Matris vurma (200×200)
   - Hash hesablaması (SHA-256, 100K dəfə)
1. **Hibrid pipeline**: I/O-dan data al → CPU ilə emal et → I/O ilə nəticəni yaz
2. `asyncio.run_in_executor()` ilə CPU tapşırıqlarını async loop-a inteqrasiya edin
3. `concurrent.futures.as_completed()` ilə nəticələri gəldikcə emal edin
4. Hər tapşırıq üçün `Future` obyektini izləyin: `done()`, `result()`, `exception()`
5. **Exception handling**: hər coroutine-də `try/except/finally`, `asyncio.gather(return_exceptions=True)`

Initial Data:

```
io_tasks = [
    {"type": "api_call", "url": f"https://jsonplaceholder.typicode.com/posts/{i}"} for i in range(1, 11)
]
cpu_tasks = [
    {"type": "factorial", "n": 50000},
    {"type": "matrix_multiply", "size": 200},
    {"type": "hash_compute", "data": "benchmark_data", "iterations": 100000},
    {"type": "prime_check", "n": 999999937},
    {"type": "fibonacci", "n": 35},
]
```

Nəticə hesabatı:

```
Metod              I/O (san)  CPU (san)  Ümumi

Ardıcıl              8.50       6.20     14.70
Asyncio only         1.20       6.18      7.38
ProcessPool only     8.45       1.85     10.30
Hibrid (async+PP)    1.20       1.85      3.05
```

Assignment 2.2 — Asinxron Rate Limiter və Retry Sistemi

API çağırışları üçün rate limiting, retry, circuit breaker pattern-lərini implement edin.

Tələblər:

1. `RateLimiter` sinfi — Token Bucket alqoritmi ilə:
   - `max_tokens`, `refill_rate` (token/saniyə)
   - `async def acquire()` — token al, yoxdursa gözlə
   - `async def __aenter__` / `__aexit__` — context manager
1. `RetryPolicy` sinfi:
   - `max_retries`, `backoff_factor`, `retryable_exceptions`
   - `async def execute(coro_func, *args)` — retry məntiqi
   - Exponential backoff: 1s → 2s → 4s
   - Jitter əlavə edin (təsadüfi gecikdirmə)
1. `CircuitBreaker` sinfi:
   - Hallar: `CLOSED` (normal) → `OPEN` (bloklanmış) → `HALF_OPEN` (test)

- Ardıcıl N xəta → OPEN (30 san gözlə) → HALF_OPEN (1 test) → CLOSED/OPEN
- `async def call(coro_func, *args)`

1. Hamısını birləşdirin:

```
limiter = RateLimiter(tokens=5, refill_rate=2)
retry = RetryPolicy(max_retries=3, backoff_factor=2)
breaker = CircuitBreaker(failure_threshold=3, recovery_time=30)

async def safe_api_call(url):
    async with limiter:
        return await breaker.call(retry.execute, fetch, url)
```

1. Simulyasiya: 30 sorğu göndərin, bəziləri bilərəkdən uğursuz olsun (`httpbin.org/status/500`)
2. Nəticə: neçəsi uğurlu, neçəsi retry olundu, circuit breaker neçə dəfə açıldı

Test URL-ləri:

```
test_urls = (
    [f"https://httpbin.org/get?id={i}" for i in range(15)] +         # uğurlu
    [f"https://httpbin.org/status/500" for _ in range(5)] +           # server xətası
    [f"https://httpbin.org/delay/10" for _ in range(3)] +             # timeout
    [f"https://httpbin.org/status/429" for _ in range(4)] +           # rate limited
    ["https://invalid-domain-xyz.com/api" for _ in range(3)]          # DNS xətası
)
```

Assignment 2.3 — Asinxron Event-Driven Sistem

Publish-Subscribe pattern ilə asinxron hadisə sistemi yazın.

Tələblər:

1. `EventBus` sinfi:
   - `subscribe(event_type, callback)` — asinxron callback qeydiyyatı
   - `publish(event_type, data)` — hadisə yayınla, bütün subscriber-ləri `asyncio.gather` ilə çağır
   - `unsubscribe(event_type, callback)`
   - Wildcard dəstəyi: `subscribe("order.*", handler)` — "order.created", "order.shipped" və s. hamısını tutsun
1. `EventStore` — bütün hadisələri tarixçə olaraq saxlasın (async file yazma)
2. **Subscriber-lər** (hər biri async):
   - `async def email_notifier(event)` — "email göndərir" (simulyasiya)
   - `async def inventory_updater(event)` — inventarı yeniləyir
   - `async def analytics_logger(event)` — analitika log-layır
   - `async def fraud_detector(event)` — şübhəli əməliyyatları yoxlayır
1. Xəta halları: subscriber xəta versə, digərləri davam etsin (`return_exceptions=True`)
2. `asyncio.wait_for` ilə subscriber timeout-u (5 san)

Initial Data — simulyasiya ediləcək hadisələr:

```
events_stream = [
    {"type": "order.created", "data": {"id": "ORD-501", "customer": "Əli", "total": 250.00, "items": ["Laptop çantası", "USB
hub"]}},
    {"type": "payment.received", "data": {"order_id": "ORD-501", "amount": 250.00, "method": "card"}},
    {"type": "order.shipped", "data": {"id": "ORD-501", "carrier": "AzPost", "tracking": "AZ123456"}},
    {"type": "user.registered", "data": {"id": "U-100", "name": "Nigar", "email": "nigar@mail.az"}},
    {"type": "order.created", "data": {"id": "ORD-502", "customer": "Tural", "total": 15000.00, "items": ["Gaming PC"]}},
    {"type": "payment.failed", "data": {"order_id": "ORD-502", "reason": "insufficient_funds"}},
    {"type": "order.cancelled", "data": {"id": "ORD-502", "reason": "payment_failed"}},
    {"type": "order.created", "data": {"id": "ORD-503", "customer": "Leyla", "total": 89.99, "items": ["Qulaqlıq"]}},
    {"type": "payment.received", "data": {"order_id": "ORD-503", "amount": 89.99, "method": "cash"}},
    {"type": "user.login", "data": {"id": "U-100", "ip": "10.0.0.55", "attempts": 7}},
    {"type": "order.delivered", "data": {"id": "ORD-501", "signed_by": "Əli Həsənov"}},
    {"type": "order.returned", "data": {"id": "ORD-503", "reason": "defective", "refund": 89.99}},
]
```

Bonus Assignment 2.B — Refaktor: Database İnteqrasiyası

> *Gün 3-ün mövzularını öyrəndikdən sonra edin.*

Assignment 2.3 (Event System) kodunu refaktor edin:

1. `EventStore`-u SQLite database-ə köçürün — `aiosqlite` ilə asinxron DB yazma
2. Hadisələri DB-dən query edən `async def get_events(event_type, date_range)` yazın
3. Subscriber nəticələrini DB-yə saxlayın (audit trail)
4. DB əməliyyatlarını `concurrent.futures` ilə ayrı thread-də icra edin

Gün 3 — Integration and Deployment

Mövzular: Database inteqrasiyası, packaging, distribution, Docker

---

Assignment 3.1 — SQLite İnventar Sistemi

Həftə 1-dən tanış olan inventar konsepsiyasını bu dəfə verilənlər bazası ilə tam implement edin.

Tələblər:

1. `sqlite3` ilə database yaradın, cədvəllər:

```
CREATE TABLE products (sku TEXT PK, name TEXT, price REAL, qty INT, category TEXT, supplier TEXT);
CREATE TABLE transactions (id INTEGER PK AUTOINCREMENT, sku TEXT FK, action TEXT, qty INT, timestamp TEXT, user TEXT);
CREATE TABLE suppliers (id TEXT PK, name TEXT, contact TEXT, rating REAL);
CREATE TABLE categories (name TEXT PK, description TEXT, tax_rate REAL);
```

1. CRUD əməliyyatları: `add_product()`, `update_product()`, `delete_product()`, `get_product(sku)`
2. Əməliyyat funksiyaları: `sell(sku, qty)`, `restock(sku, qty)` — transaction cədvəlinə yazılsın
3. Query funksiyaları:
   - Kateqoriyaya görə ümumi dəyər (`SUM(price * qty) GROUP BY category`)
   - Ən çox satılan 5 məhsul (`JOIN transactions ... ORDER BY ...`)
   - Stoku bitmək üzrə olanlar (`WHERE qty < threshold`)
   - Təchizatçıya görə statistika
1. `context manager` ilə DB bağlantısı: `with DatabaseManager("inventory.db") as db:`
2. SQL injection-dan qorunma: parametrized queries
3. Migration sistemi: versiyalı schema dəyişiklikləri
4. Nəticəni formatla:

```
 İNVENTAR HESABATI

Ümumi məhsul: 45 | Ümumi dəyər: 28,450 AZN
Stoku bitmiş: 3 | Aşağı stok: 7

Kateqoriya      Məhsul  Dəyər (AZN)

Elektronika       12     15,200
Aksessuar         18      8,350
```

Initial Data (DB-yə yüklənəcək):

```
initial_products = [
    ("SKU001", "Mexanik Klaviatura", 89.99, 45, "Elektronika", "SUP-001"),
    ("SKU002", "Simsiz Siçan", 34.50, 120, "Elektronika", "SUP-001"),
    ("SKU003", "USB-C Hub", 52.00, 0, "Aksessuar", "SUP-002"),
    ("SKU004", "Monitor Standı", 45.00, 18, "Aksessuar", "SUP-003"),
    ("SKU005", "Noutbuk Çantası", 29.99, 200, "Aksessuar", "SUP-004"),
    ("SKU006", "Webcam HD", 75.00, 8, "Elektronika", "SUP-002"),
    ("SKU007", "Qulaqlıq", 120.00, 32, "Elektronika", "SUP-005"),
    ("SKU008", "SSD 1TB", 95.00, 15, "Elektronika", "SUP-001"),
    ("SKU009", "Ergonomik Kreslo", 350.00, 5, "Mebel", "SUP-003"),
    ("SKU010", "Ağ Taxta", 180.00, 10, "Mebel", "SUP-003"),
]

initial_suppliers = [
    ("SUP-001", "TechCo", "info@techco.az", 4.5),
    ("SUP-002", "GadgetPro", "sales@gadgetpro.com", 4.2),
    ("SUP-003", "OfficePlus", "order@officeplus.az", 3.8),
    ("SUP-004", "BagWorld", "contact@bagworld.az", 4.0),
    ("SUP-005", "AudioMax", "support@audiomax.com", 4.7),
]
```

Assignment 3.2 — Python Paketləmə və Distribution

Əvvəlki tapşırıqlarda yazdığınız modullardan birini (və ya yenisini) tam paketlənmiş, quraşdırıla bilən Python paketi halına gətirin.

Yaradılacaq paket: `datakit` — data emal utility-ləri

Paket strukturu:

```
datakit/
 pyproject.toml
 README.md
 LICENSE
 src/
    datakit/
        __init__.py
        csv_tools.py      # CSV oxuma/yazma/filtrləmə
        json_tools.py     # JSON emal
        stats.py          # Statistik hesablamalar
        validators.py     # Data validasiyası
        cli.py            # CLI entry point
 tests/
    __init__.py
    test_csv_tools.py
    test_json_tools.py
    test_stats.py
    test_validators.py
 examples/
     usage_example.py
```

Tələblər:

1. `pyproject.toml` konfiqurasiyası: metadata (ad, versiya, müəllif, təsvir), dependencies, entry_points
2. `src` layout istifadə edin (modern Python packaging)
3. CLI entry point: `[project.scripts] datakit = "datakit.cli:main"` — quraşdırıldıqdan sonra `datakit` əmri ilə çalışsın
4. Funksionallıq:
   - `csv_tools`: `read_csv`, `write_csv`, `filter_rows`, `merge_csvs`, `pivot`
   - `json_tools`: `flatten_json`, `merge_jsons`, `json_to_csv`, `validate_schema`
   - `stats`: `describe`, `correlate`, `detect_outliers`, `moving_average`
   - `validators`: `validate_email`, `validate_date`, `validate_range`, `validate_schema`
1. `unittest` ilə test yazın (minimum 3 test hər modul üçün)
2. `pip install -e .` ilə development mode-da quraşdırın
3. `README.md` — istifadə nümunələri, API sənədləşdirməsi

Assignment 3.3 — Docker ilə Konteynerləşdirmə

Assignment 3.1 (İnventar Sistemi) və ya Assignment 3.2 (datakit) paketini Docker konteyneri olaraq paketləyin.

Tələblər:

1. **Dockerfile** yazın:

```
# Multi-stage build
FROM python:3.12-slim AS builder
WORKDIR /app
COPY pyproject.toml .
COPY src/ src/
RUN pip install --no-cache-dir .

FROM python:3.12-slim
WORKDIR /app
COPY --from=builder /usr/local/lib/python3.12/site-packages /usr/local/lib/python3.12/site-packages
COPY --from=builder /usr/local/bin/datakit /usr/local/bin/datakit
# ...
```

1. **docker-compose.yml** — əgər DB istifadə edirsinizsə:

```
services:
  app:
    build: .
    volumes:
      - ./data:/app/data
    environment:
      - DB_PATH=/app/data/inventory.db

  # opsional: web interfeys üçün
  web:
    build: ./web
    ports:
      - "8080:8080"
    depends_on:
      - app
```

1. `.dockerignore` faylı: `__pycache__`, `.git`, `*.pyc`, `.env`
2. **Environment variables** ilə konfiqurasiya: `DB_PATH`, `LOG_LEVEL`, `API_KEY`
3. **Health check**: konteynerin sağlam olub-olmadığını yoxlayan endpoint/script
4. **Volume** ilə persistent data: DB faylı konteyner yenidən başladıqda itməsin
5. `docker build -t datakit:1.0 .` və `docker run` əmrləri ilə test edin
6. Multi-stage build ilə final image ölçüsünü minimuma endirin
7. Sənədləşdirmə: README-yə Docker ilə istifadə təlimatı əlavə edin

---

Həftəlik Assignment — Asinxron Xəbər Aqreqasiya Platforması

> *Bütün həftənin mövzularını əhatə edir: asyncio, async/await, futures, concurrent.futures, exception handling, database, packaging, Docker.*

Sistemin Təsviri

Müxtəlif mənbələrdən asinxron data toplayan, SQLite-da saxlayan, analiz edən və Docker-da deploy olunan tam platforma yazın.

Paket strukturu:

```
news_platform/
 pyproject.toml
 Dockerfile
 docker-compose.yml
 .env.example
 README.md
 src/
    news_platform/
        __init__.py
        core/
           __init__.py
           collector.py     # Asinxron data toplayıcı
           processor.py     # Data emalçısı
           scheduler.py     # Vaxtlanmış tapşırıqlar
        sources/
           __init__.py
           base.py          # Base source sinfi
           api_source.py    # API mənbələri
           scraper_source.py # Scraping mənbələri
        storage/
           __init__.py
           database.py      # SQLite əməliyyatlar
           models.py        # DB modellər
        analysis/
           __init__.py
           trending.py      # Trend analizi
           stats.py         # Statistika
        resilience/
           __init__.py
           rate_limiter.py
           retry.py
           circuit_breaker.py
        cli.py
        config.py
 tests/
    test_collector.py
    test_database.py
    test_resilience.py
 data/
     .gitkeep
```

Initial Data — mənbə konfiqurasiyaları:

```
sources_config = {
    "hacker_news": {
        "type": "api",
        "base_url": "https://hacker-news.firebaseio.com/v0",
        "endpoints": {
            "top": "/topstories.json",
            "item": "/item/{id}.json"
        },
        "rate_limit": {"requests": 10, "per_seconds": 60},
        "refresh_interval": 300
    },
    "github_trending": {
        "type": "scraper",
        "url": "https://github.com/trending",
        "selectors": {
            "repo": "article.Box-row",
            "name": "h2 a",
            "description": "p.col-9",
            "language": "span[itemprop='programmingLanguage']",
            "stars": "a.Link--muted"
        },
        "refresh_interval": 600
    },
    "jsonplaceholder": {
        "type": "api",
        "base_url": "https://jsonplaceholder.typicode.com",
        "endpoints": {
            "posts": "/posts",
            "users": "/users"
        },
        "rate_limit": {"requests": 30, "per_seconds": 60},
        "refresh_interval": 120
    },
    "nasa_apod": {
        "type": "api",
        "base_url": "https://api.nasa.gov",
        "endpoints": {
            "apod": "/planetary/apod"
        },
        "params": {"api_key": "DEMO_KEY"},
        "rate_limit": {"requests": 30, "per_seconds": 3600},
        "refresh_interval": 86400
    }
}
```

Tələblər

Hissə 1 — Asinxron Data Toplama (asyncio, async/await)

1. `BaseSource` sinfi: `async def fetch()`, `async def parse(response)`, `async def validate(data)`
2. `APISource(BaseSource)`: `aiohttp` ilə API sorğuları
3. `ScraperSource(BaseSource)`: HTML parse (aiohttp + BeautifulSoup)
4. `Collector` sinfi: `asyncio.gather()` ilə bütün mənbələrdən paralel data toplasın
5. `asyncio.Semaphore` ilə mənbə başına limit
6. `Scheduler`: `asyncio.create_task` + `asyncio.sleep` ilə vaxtlanmış yeniləmə

Hissə 2 — Resilience Patterns (futures, exception handling)

1. `RateLimiter` — token bucket, mənbə başına ayrı limit
2. `RetryPolicy` — exponential backoff + jitter, `return_exceptions=True`
3. `CircuitBreaker` — CLOSED/OPEN/HALF_OPEN, failure threshold, recovery
4. Bütün async xətaları düzgün idarə olsun: `TimeoutError`, `aiohttp.ClientError`, `json.JSONDecodeError`
5. `concurrent.futures.ProcessPoolExecutor` ilə CPU-intensive analiz (trend, statistika)

Hissə 3 — Database (SQLite)

1. Cədvəllər: `articles`, `sources`, `fetch_logs`, `trends`
2. CRUD: `insert_article`, `get_articles(filters)`, `update_article`, `mark_as_read`
3. Analiz query-ləri: mənbəyə görə statistika, saatlıq trend, ən populyar mövzular
4. `asyncio.run_in_executor()` ilə DB əməliyyatlarını async loop-a inteqrasiya edin

Hissə 4 — Packaging və Docker

1. `pyproject.toml` ilə paketləyin, CLI entry point: `newsplatform collect`, `newsplatform report`, `newsplatform stats`
2. `Dockerfile` (multi-stage build), `docker-compose.yml` (volume ilə DB persist)
3. `.env` ilə API key-lər, `config.py` ilə oxuma
4. `unittest` ilə testlər (minimum 10 test)

Nəticə Hesabatı

1. CLI hesabat çıxışı:

```
       XƏBƏR PLATFORMASI HESABATI — 2026-03-17


 Toplama Statistikası
   Son yeniləmə: 2026-03-17 22:30
   Ümumi məqalə: 127 | Yeni (bu gün): 34
   Aktiv mənbə: 4/4 | Xətalar: 2

 Mənbə Üzrə
Mənbə            Məqalə  Son Yeniləmə   Status

Hacker News        30    5 dəq əvvəl
GitHub Trending    25    10 dəq əvvəl
JSONPlaceholder    50    2 dəq əvvəl
NASA APOD           1    bu gün

  Trend Mövzular
1. "AI" — 12 dəfə | ↑ 40% bu həftə
2. "Rust" — 8 dəfə | ↑ 25%
3. "Python" — 7 dəfə | → sabit

  Resilience Statistikası
Rate Limit: 3 gözlədildi | Retry: 5 uğurlu
Circuit Breaker: 0 açılma | Timeout: 2
```