**2. URL Shortener (Redis-Centric)**

**Objective**

Create a high-performance URL shortening service with click tracking.

**Tech Stack**

* Django (Web Layer)
* Redis (Primary Data Store)
* PostgreSQL (Optional for backup)

**Database Architecture**

python

Copy

# Redis Key Structure

SHORT\_CODE\_MAPPING = "url:{short\_code}" # String type

CLICK\_COUNTER = "clicks:{short\_code}" # Integer type

CACHE\_KEY = "cache:url:{short\_code}" # String type with TTL

**Project Architecture**

Copy

- Django App

├── URL Converter (Base62)

├── Redis Manager

└── Analytics Service

- Redis

├── URL Mappings

├── Click Counters

└── Cache Layer

**Implementation Steps**

1. **Short URL Generation**
   * Use Base62 encoding (a-zA-Z0-9) for short codes
   * Generate 7-character codes (62^7 = ~3.5 trillion combinations)

python

Copy

def generate\_short\_code():

return ''.join(choices(ascii\_letters + digits, k=7))

1. **Redis Operations**

python

Copy

def create\_short\_url(original\_url):

short\_code = generate\_short\_code()

# Store mapping with EX (expiry) if needed

redis.setex(f'url:{short\_code}', 3600\*24\*30, original\_url) # 30 days

return short\_code

def get\_original\_url(short\_code):

# Check cache first

cached = redis.get(f'cache:url:{short\_code}')

if cached:

return cached

# Then check persistent storage

original = redis.get(f'url:{short\_code}')

if original:

# Cache for frequent access

redis.setex(f'cache:url:{short\_code}', 3600, original)

return original

1. **Click Tracking**

python

Copy

def track\_click(short\_code):

redis.incr(f'clicks:{short\_code}')

redis.expire(f'clicks:{short\_code}', 3600\*24\*30) # Match URL expiry

1. **API Endpoints**
   * POST /api/shorten/ - Create short URL
   * GET /{short\_code} - Redirect with 301/302
   * GET /api/analytics/{short\_code} - Get click count

**Advanced Features**

* Rate limiting using Redis INCR + EXPIRE
* Bulk URL creation
* User-specific URL management
* Expiring URLs with TTL

**Do's/Don'ts**

* ✅ **Do**: Use Redis transactions (MULTI/EXEC) for critical operations
* ✅ **Do**: Implement LRU cache eviction policy
* ❌ **Don't**: Store user sessions in same Redis instance
* ❌ **Don't**: Use Redis keys without expiry