Backend Assignment: URL Shortener Service

Objective:
Design and implement a production-ready backend service that allows users to shorten long URLs and retrieve the original URLs efficiently.

Requirements

1. **Core Functionality**
   - Users should be able to:
     - Submit a long URL and receive a shortened version (e.g., `https://sho.rt/abc123`).
     - Access the shortened URL and get redirected to the original long URL.
   - Each shortened URL should be unique and no longer than **10 characters**.
   - The same original URL should always return the same short code (idempotent).
2. **Data Handling**
   - Store mappings between `short_code` and `original_url`.
   - Record metadata such as:
     - `created_at`
     - `click_count`
     - `last_accessed_at`
3. **Constraints**
   - Handle at least **10,000 new URLs per day**.
   - URLs must be stored for **at least 5 years**.
   - Redirect requests should complete in **under 100 ms**.

    - Handle URL collisions and duplicates gracefully.
    - Validate input (only valid URLs).
    - Implement basic **rate limiting** (e.g., per IP).
    - Include **pagination** for listing all shortened URLs (for admin use).

Technical Requirements

- **Language:** Go / Python / Node.js
- **Database:** PostgreSQL or Redis (or both)
- **Architecture:**
    - RESTful API (bonus for gRPC or GraphQL)
    - Clean, modular folder structure (e.g., handler → service → repository)
    - Environment-based configuration using `.env`
    - Graceful shutdown and proper error handling

Containerization

- Create a **Dockerfile** that:
    - Builds and runs the service in a lightweight image (e.g., Alpine-based)
    - Exposes the service on port **8080**
    - Uses multi-stage builds for smaller image size
- Add a **docker-compose.yml** to:
    - Spin up the backend service and database together
    - Auto-load environment variables from `.env`

Ensure the application can be started using:

```
docker-compose up --build
```

-
- The final image should be **publishable** (e.g., `docker.io/<username>/url-shortener:latest`)

Bonus Features (Optional)

- Caching layer (Redis) to reduce redirect latency
- Analytics endpoint (e.g., total clicks per day)
- Token-based authentication for admin routes
- Support for custom aliases (`/mybrand`)
- CI/CD workflow (GitHub Actions or similar)

Deliverables

- GitHub repository containing:
    - Complete source code
    - `Dockerfile` and `docker-compose.yml`
    - README with:
        - Setup & run instructions
        - API documentation with sample cURL or Postman requests
        - Architectural overview (diagram optional)
        - Design decisions and trade-offs