ChatGPT

URL Shortener Service

This is a FastAPI-based URL shortener service that allows users to create short links for long URLs, track click counts, and fetch statistics for each short link. The project uses PostgreSQL as the database and is fully containerized using Docker and Docker Compose. It includes automated testing and continuous integration via GitHub Actions.

Table of Contents

Features

- Create a short code for a target URL (idempotent: returns existing code if URL already shortened)
- Redirect from the short URL to the original URL
- Track and increment click counts for each short URL
- Fetch total click stats for a short URL (requires API key)
- Automated tests with pytest
- Continuous integration using GitHub Actions
- Containerization with Docker and Docker Compose

Prerequisites

- Python 3.11 or newer
- Git
- Docker & Docker Compose (if running with containers)
- PostgreSQL (if running locally without Docker)

Project Structure

```
url-shortener/
├── alembic/                  # Alembic migrations
├── app/
│   ├── routers/              # FastAPI routers
│   ├── models.py             # SQLAlchemy models
│   ├── schemas.py            # Pydantic schemas
│   ├── database.py           # SQLAlchemy engine & session
│   ├── config.py             # Environment variable loading
│   ├── main.py               # FastAPI application setup
│   └── create_db.py          # Optional script to create DB tables
├── tests/                    # Test suite (pytest)
│   ├── test_main.py          # Tests for API endpoints
│   └── test_models.py        # Tests for database models
├── .github/workflows/        # GitHub Actions workflows
│   └── ci.yml                # CI configuration file
├── Dockerfile                # Container image definition
├── docker-compose.yml        # Docker Compose configuration
├── requirements.txt          # Python dependencies
├── alembic.ini               # Alembic configuration
├── README.md                 # Project documentation
└── .env                      # Environment variables (not committed)
```

Environment Variables

Copy the `.env.example` (if provided) or create a `.env` file in the root directory with the following:

```
DATABASE_URL=postgresql://<DB_USER>:<DB_PASSWORD>@db:5432/<DB_NAME>
API_KEY=<your_api_key>
```

- `DATABASE_URL`: Connection string for PostgreSQL. In Docker setup, use `db` as hostname (service name). - `API_KEY`: A token to access the `/stats/{short_code}` endpoint.

Example:

```
DATABASE_URL=postgresql://user:password@db:5432/shortener
API_KEY=Rawan-711
```

Local Setup

Clone the Repository

```
git clone https://github.com/SalakAltaf/url-shortener.git
cd url-shortener
```

Python Virtual Environment

```
python -m venv venv
# On Windows:
venv\Scripts\activate
# On Linux/macOS:
source venv/bin/activate
```

Install Dependencies

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

Database Setup (Local PostgreSQL)

1. Ensure PostgreSQL is installed and running.
2. Create a database and user matching your `DATABASE_URL`:

   ```
   CREATE DATABASE shortener;
   CREATE USER user WITH ENCRYPTED PASSWORD 'password';
   GRANT ALL PRIVILEGES ON DATABASE shortener TO user;
   ```

3. Update `.env` with the correct connection string if not using Docker.

Run Migrations

```
alembic upgrade head
```

Run the Application

```
uvicorn app.main:app --reload
```

Visit `http://localhost:8000` in your browser. The API docs are available at `http://localhost:8000/docs`.

Docker Setup (Recommended)

Build and Run with Docker Compose

Make sure Docker and Docker Compose are installed.

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

- This will start two containers: - `db`: PostgreSQL database - `url-shortener-app`: FastAPI application

Accessing the App

- Browse to `http://localhost:8000` to see the FastAPI welcome or docs.
- API docs (Swagger UI) at `http://localhost:8000/docs`.

Stopping the Containers

```
docker-compose down
```

This stops and removes the containers (but retains database data in Docker volume).

Running Tests

With the virtual environment active:

```
pytest
```

To run only specific tests, specify the file:

```
pytest tests/test_main.py
```

In CI, tests are run automatically via GitHub Actions.

API Documentation

Shorten a URL

Request: `POST /shorten` - **Headers**: None - **Body** (JSON):

```
{
  "url": "https://example.com"
}
```

Response (201):

```
{
  "code": "abc123",
  "short_url": "http://<host>:<port>/abc123"
}
```

If the URL was already shortened, it returns the existing code.

Redirect Short URL

Request: `GET /{short_code}` - **Path Param**: `short_code` (the code returned by `/shorten`).

Behavior: Redirects (HTTP 307) to the original target URL and increments click count.

Get URL Stats

Request: `GET /stats/{short_code}` - **Path Param**: `short_code` - **Headers**: - `token`: `<API_KEY>`

Response (200):

```
{
  "short_code": "abc123",
  "clicks": 42
}
```

If the `token` header is missing or invalid, returns 401 or 403.

GitHub Actions CI

Every push to `master` runs the following workflow: 1. **Checkout code** 2. **Set up Python** (3.11) 3. **Install dependencies** via `pip install -r requirements.txt` 4. **Start PostgreSQL** as a service (using `postgres:15` image) 5. **Run** `pytest` with `DATABASE_URL` and `API_KEY` set 6. **Build Docker image** to ensure it builds correctly

You can view status in the Actions tab of the GitHub repository.

License

This project is licensed under the MIT License. See LICENSE for details.