



Experiment 3

Student Name: Sujal Srivastava

Branch: BE-CSE

Semester: 6th

UID: 23BCS11842

Section/Group: KRG-3A

Date of Performance: 07-01-26

Subject Name: SYSTEM DESIGN

Subject Code: 23CSH-314

1. Aim: To study and design a scalable URL Shortener system by analyzing its functional requirements, non-functional requirements, API design, database schema, and low-level design approaches including counter-based short URL generation for distributed systems.

2. Objective:

1. To understand the working of a URL shortening service.
2. To design REST APIs for URL creation and redirection.
3. To identify suitable database and server choices.
4. To analyze different short URL generation techniques.
5. To understand scalability challenges and their solutions.

3. REQUIREMENTS:

1. Functional Requirements

The system should generate a short, unique URL for a given long URL.

When a user accesses the short URL, the system should redirect to the original long URL.

The system should support millions of URLs and optionally track click counts and expiration.

2. Non-Functional Requirements

The system must be highly available and low latency, as redirection should be very fast.

It should be scalable to handle a large number of users and URLs.

The system must ensure uniqueness, reliability, and data consistency.

3. API Design (URL Shortener)

POST /shorten → Accepts a long URL and returns a generated short URL.

GET /{shortUrl} → Redirects the user to the original long URL.

Optional APIs can support analytics, deletion, and expiration of URLs.

4. High Level Design (HLD) of URL Shortener

The system consists of a client, load balancer, application server, cache, and database.

The application server generates a short code, stores the mapping in the database, and uses cache for fast redirection.

Load balancers distribute traffic, while the database ensures durable storage.

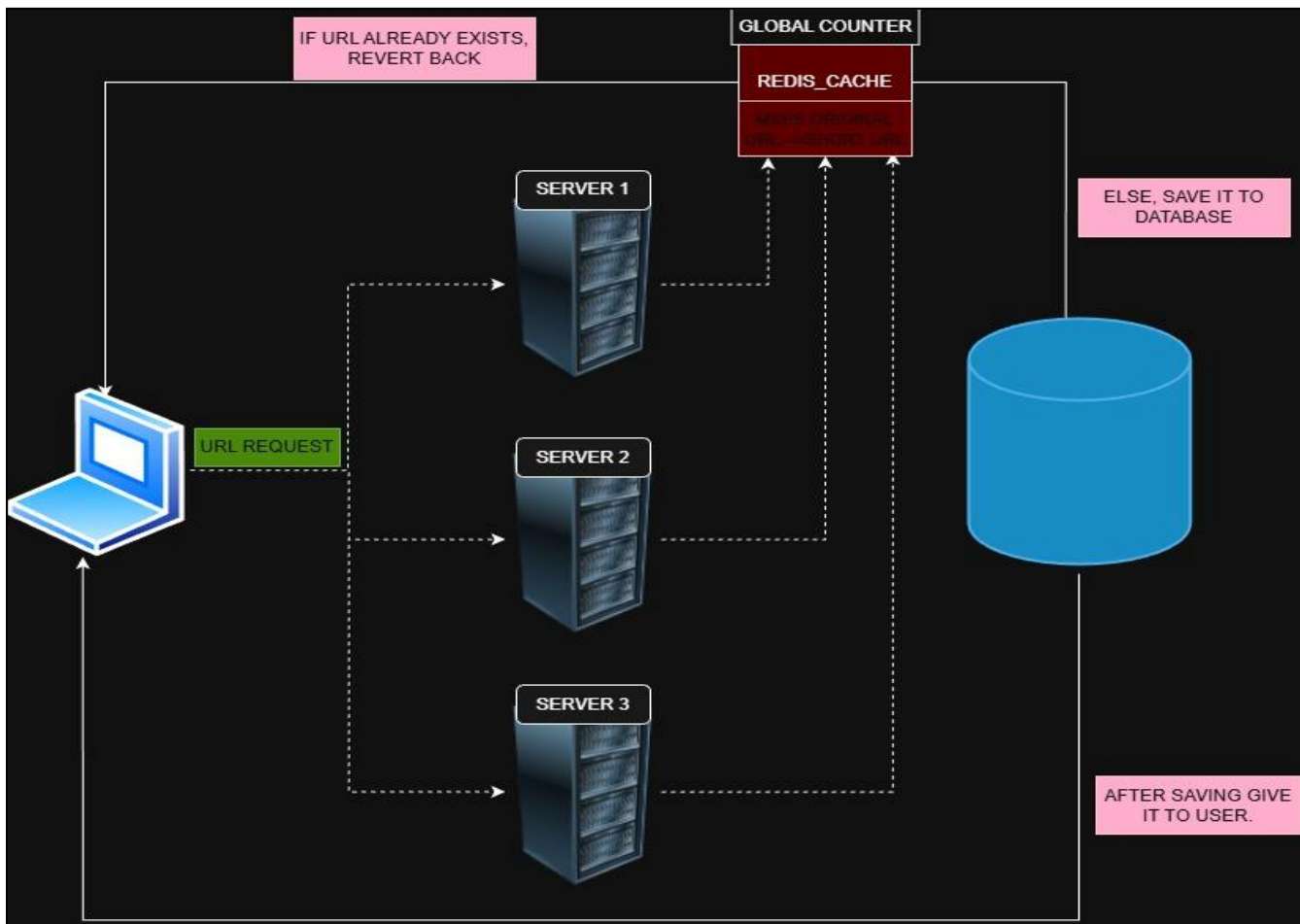
5. Low Level Design (LLD) of URL Shortener

The system uses a hashing or ID-encoding technique (Base62) to generate unique short URLs.

A database table stores mappings like (shortKey, longURL, createdAt, expiry, clickCount).

On redirection, the server first checks the cache, then the database, and returns an HTTP 301/302 redirect.

4. DIAGRAMS:



5. Learning Outcome

- This project helped in understanding the end-to-end design of a real-world system, including requirement analysis, API design, and system architecture.
- It provided practical knowledge of scalability, database design, caching, and performance optimization concepts.
- The project also improved skills in problem solving, backend logic development, and applying data structures in system design.