



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

**Student Name:** Shashwat Mishra  
**Branch:** CSE  
**Semester:** 6th  
**Subject Name:** System Design

**UID:** 23BCS10866  
**Section/Group:** KRG 1A  
**Date of Performance:** 11/01/2026  
**Subject Code:** 23CSH-314

**1. Aim:** To design and analyze a **URL Shortener System** that converts long URLs into short, unique URLs while ensuring high availability, scalability, low latency, and efficient redirection. The system also supports optional custom URLs, expiration dates, and user authentication.

## 2. Objective:

- To understand functional and non-functional requirements of a large-scale distributed system.
- To design RESTful APIs for URL creation and redirection.
- To identify core entities such as User, Short URL, and Long URL.
- To analyse CAP theorem trade-offs and apply eventual consistency.
- To design high-level and low-level architecture for a scalable URL shortener.
- To study multiple approaches for short URL generation and compare their performance.

## 3. Tools Used:

- **Python** – Backend logic implementation and URL generation algorithms.
- **Flask** – Lightweight web framework for developing RESTful APIs.
- **Draw.io** – Designing system architecture diagrams (HLD & LLD).

## 4. System Requirements:

### A. Functional Requirements

- Create a short URL from a given long URL.
- Support optional custom short URLs.
- Support default and user-defined expiration dates.
- Redirect users from short URL to the original long URL.
- Provide REST APIs for URL creation and redirection.
- Support user registration and login using REST APIs.

# **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

## **B. Non-Functional Requirements**

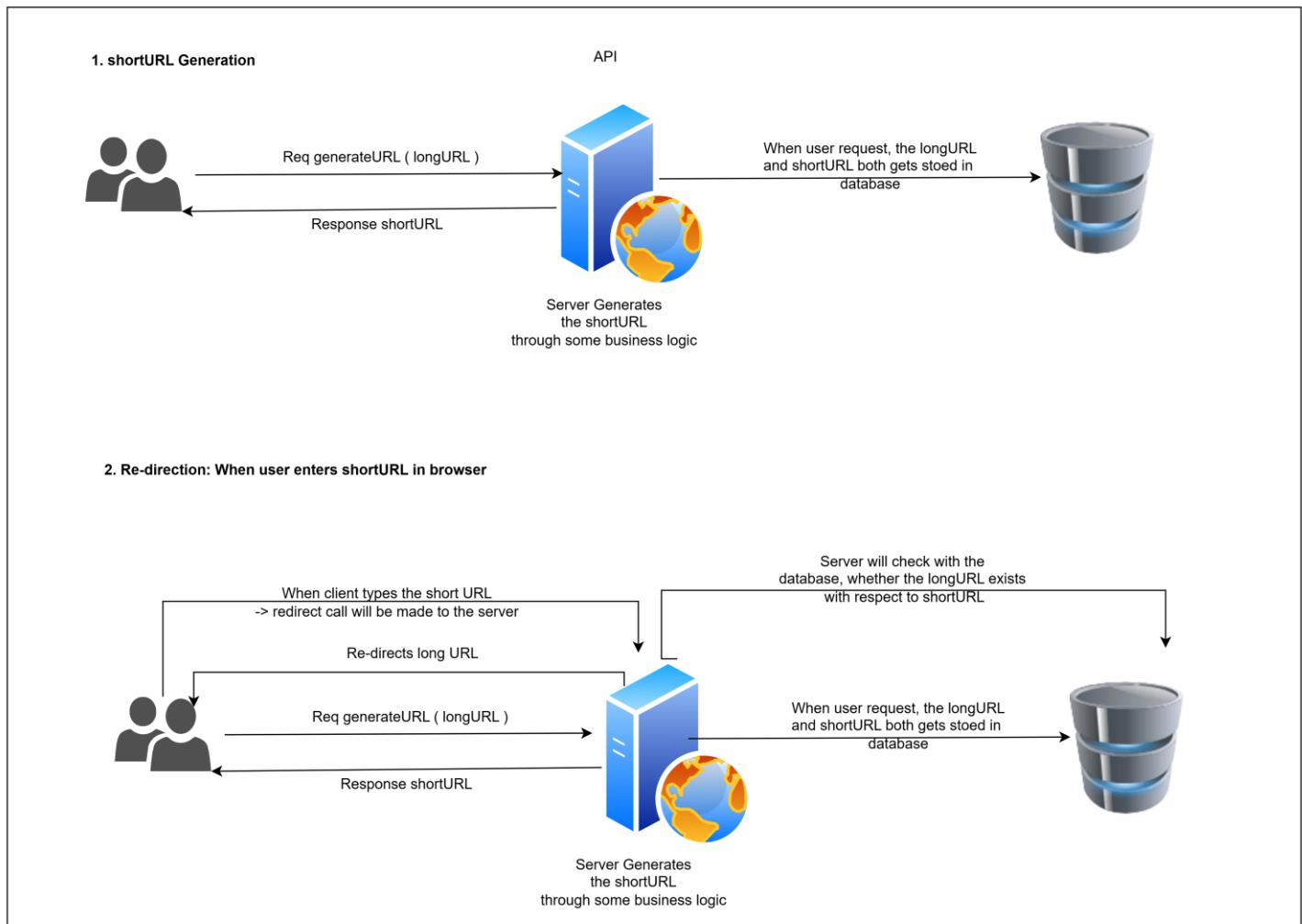
- Low latency ( $\leq 200$  ms for URL creation and redirection).
- High scalability (100M daily active users, 1B URLs).
- High availability (24x7).
- Uniqueness of short URLs.
- High availability preferred over strict consistency (Eventual Consistency).

## **5. High Level Design (HLD):**

The system follows a **Client–Server–Database architecture**:

- Client sends request to generate or access short URL.
- Server processes business logic and generates short URL.
- Database stores mappings of short URL and long URL.
- On redirection, server fetches long URL and redirects the user.

### 1. High Level Design



# **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

## **6. Low Level Design (LLD):**

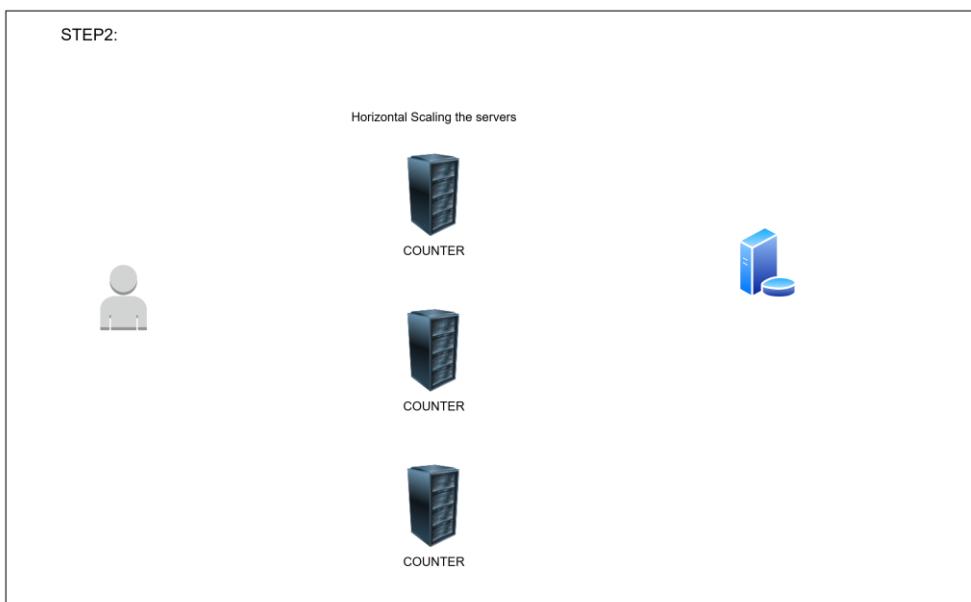
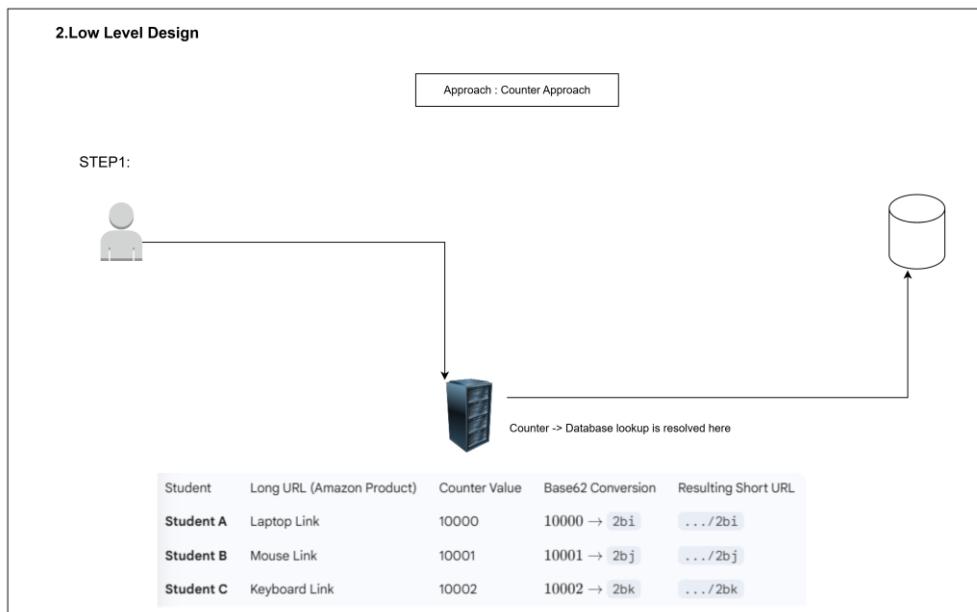
**Approach :** Counter-Based

- Uses auto-increment counter.
- Counter value converted to Base62 for short URL.

Example:

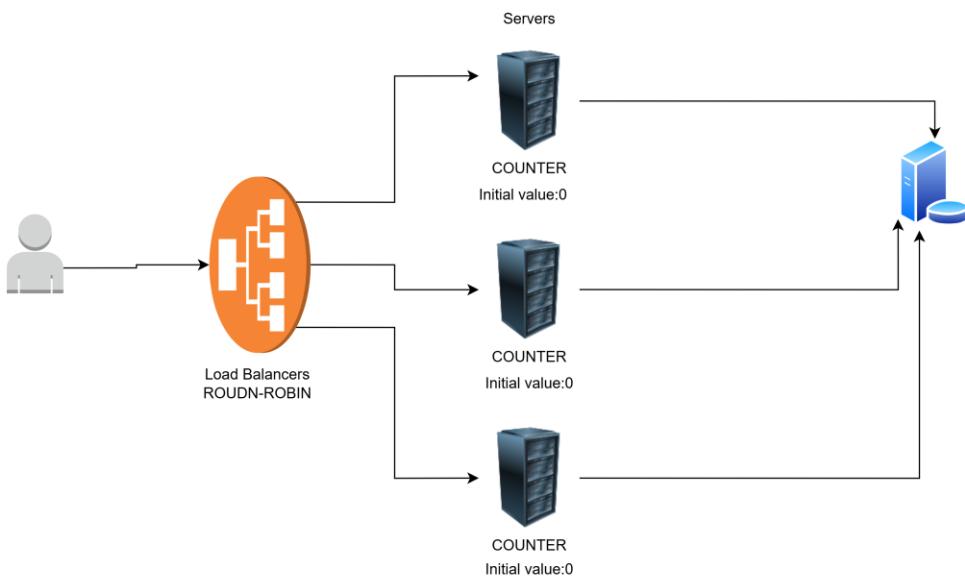
Counter: 10000 → Base62: 2bi → Short URL

- Issue: Single counter causes scalability issues.

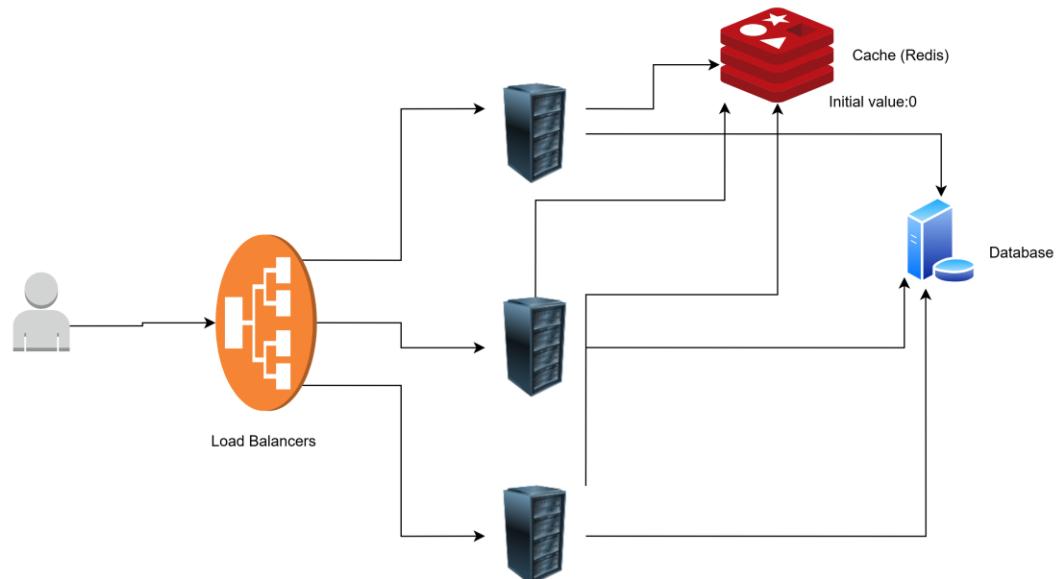


# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

STEP3:



STEP4:



## 7. Scalability Solution

- Horizontal scaling of application servers.
- Use of Load Balancer (Round Robin).

# **DEPARTMENT OF**

# **COMPUTER SCIENCE & ENGINEERING**

- Centralized counter stored in Redis cache.
- Redis ensures fast access and atomic increments.
- Database stores final URL mappings.

# **DEPARTMENT OF**

# **COMPUTER SCIENCE & ENGINEERING**

## **8. Learning Outcomes:**

- Learned how to design a real-world scalable system.
- Understood REST API design principles. • Gained knowledge of CAP theorem and eventual consistency.
- Learned multiple URL shortening techniques and their trade-offs.
- Understood horizontal scaling, caching, and load balancing.
- Learned importance of low latency and high availability systems.