Experiment 1

Student Name: Muskan Yadav   **UID:** 23BCS10179
Branch: BE-CSE   **Section:** Group:KRG-3B
Semester: 6   **Date of performance**: 06-01-2026
Subject name: System Design   **Subject code**: 23CSH-314

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.

Objective:

1. To understand the internal workflow of a URL shortening service.
2. To design RESTful APIs for creating short URLs and handling redirections.
3. To select appropriate server architecture and database technologies.
4. To study and compare different techniques for generating short URLs.
5. To evaluate scalability challenges and propose efficient solutions.

Problem Statement

Design a URL Shortener system capable of transforming long URLs into compact short links and efficiently redirecting users to the original URLs while ensuring minimal latency, high system availability, and seamless scalability.

Requirements

A. Functional Requirements

- Convert a given long URL into a unique short URL.

- Allow users to create custom short URLs (optional feature).

- Support URL expiration with both default and user-defined expiry times.

- Redirect requests from a short URL to its corresponding long URL.

- Provide user registration and authentication using REST APIs.

B. Non-Functional Requirements

1. Low Latency: URL generation and redirection should complete within 200 ms.

2. Scalability:

   - Support up to 100 million daily active users
   - Store and manage up to 1 billion shortened URLs

3. High Availability: System should be operational 24×7.

4. Uniqueness: Every generated short URL must be unique.

5. CAP Theorem Consideration:

   - Prioritize availability over strong consistency
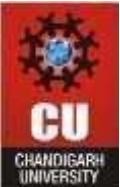   - Follow an eventual consistency model

System Entities

1. **User**

2. **Long URL**

3. **Short URL**

API Design

1. Create Short URL Endpoint: POST /v1/url

Request Body:

{

  "longURL": "string",

  "customURL": "(string optional)",

  "expirationDate": "date"

}

Response:

{

  "shortURL": "https://short.ly/2bi"

}

2. Redirect Short URL

Endpoint: GET /v1/url/{shortURL}

Action: Redirects to original long URL.

3. User APIs

- POST /v1/register

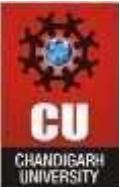- POST /v1/login

High Level Design (HLD):

Components:

- Client (Web Browser / Mobile App)
- Application Server (Business Logic Layer)
- Database (Persistent Storage)

Workflow

1. Short URL Creation

- Client submits a long URL request.

- Server generates a unique short URL.

- URL mapping is stored in the database.

- Short URL is returned to the client.

2. URL Redirection

- Client accesses the short URL.

- Server retrieves the corresponding long URL.

- Client is redirected to the original URL.

Low-Level Design Approaches:-

Approach 1: Hash-Based URL Generation

Method: Generate short URL by hashing the long URL.

Algorithms: MD5, SHA-1, Base64

Limitations:

- Hash output length is too large

- Possibility of collisions

- Increased database lookups

- Poor performance at scale

Conclusion: Not ideal for large-scale systems.

Approach 2: Counter-Based URL Generation (Preferred)

Steps:

1. Fetch a globally unique counter value.

2. Convert the counter to Base62 format.

3. Use the Base62 string as the short URL.

Example:

Counter = **125000**

Base62 = **w7E**

Short URL = **https://short.ly/w7E**

Advantages:

- No collision

- Fast generation

- Predictable length

Scalability Challenges and Solutions:-

Challenge 1: Single Server Bottleneck

Solution: Deploy multiple stateless servers with horizontal scaling.

Challenge 2: Counter Synchronization in Distributed Systems

Solution: Use an atomic global counter stored in Redis to ensure uniqueness.

Technology Choices:

Database

- **Redis** (Caching)

- **DynamoDB** (Persistent storage)

Reasons:

- High read/write performance

- High availability

- Horizontal scalability

Server Architecture

- Stateless REST API servers

- Deployed behind a load balancer

- Handles validation, counter retrieval, Base62 conversion, and database operations

Result

A scalable, high-availability URL Shortener system was designed using a counter-based Base62 encoding strategy, RESTful APIs, NoSQL databases, and distributed system principles.

Conclusion

Using a counter-based short URL generation method with centralized atomic counters ensures a fast, reliable, and collision-free solution capable of supporting millions of users with minimal latency and high availability.