

# URL Shortener – Project Report

## 1. Introduction

The internet is filled with lengthy URLs that are often difficult to share, remember, or embed within social media platforms, emails, and marketing materials. To address this challenge, the URL Shortener is a web-based tool that transforms long URLs into concise, easy-to-share links. This project implements a functional URL shortening service using Java, Spring Boot, and an H2 in-memory database, complete with a testable HTML front-end.

## 2. Abstract

The URL Shortener application enables users to input a long URL and receive a shortened version that redirects to the original address. This system uses a RESTful API built on Spring Boot and stores the mapping between original and short URLs in an H2 database. It supports browser-based redirection and provides both user-friendly and API-driven interfaces. The project showcases core backend development skills, URL mapping techniques, and full-stack integration.

## 3. Tools and Technologies Used

Tool/Technology	Purpose
Java 17	Backend programming language
Spring Boot 3.1.0	Framework for building REST APIs and services
Spring Web	To expose and consume REST endpoints
Spring Data JPA	To manage database operations
H2 Database	Lightweight, in-memory database
HTML, CSS, JavaScript	Frontend for interacting with the API
Maven	Project and dependency management

## 4. Steps Involved in Building the Project

### Step 1: Setup Maven Project

Created a Maven-based Java project using Spring Initializr.

Added dependencies: spring-boot-starter-web, spring-boot-starter-data-jpa, h2, and jakarta.persistence.

### Step 2: Model Creation

Created UrlMapping entity with fields: id, originalUrl, and shortUrl.

Used @Entity, @Id, and @GeneratedValue annotations for JPA mapping.

### Step 3: Repository Layer

Built a UrlRepository interface extending JpaRepository.

Implemented custom method findByShortUrl(String shortUrl).

### Step 4: Service Layer

Developed UrlService to handle: URL shortening by generating random codes (short.ly/123456 format).

Fetching original URL using short URL ID.

#### **Step 5:** Controller Layer

Created two controllers:

UrlController (REST API): POST /api/url/shorten, GET /api/url/{shortUrl}

RedirectController: GET /redirect/{shortCode} for browser redirection.

GET / to return a basic home page with endpoint information.

#### **Step 6:** Frontend UI

Designed a responsive HTML page (test.html) to interact with the backend.

Features:

Input field for original URL

AJAX request to shorten URL

Dynamic display of results and browser links

Clipboard copy functionality

#### **Step 7:** Testing and Debugging

Validated: API endpoint responses (JSON)

URL mapping and redirection

Error handling (invalid URLs, server not running, etc.)

Used cURL and browser to test the redirect flow.

### **5. Conclusion**

The URL Shortener project demonstrates the integration of a full-stack Java application with RESTful APIs, database interaction, and a user-friendly frontend. It emphasizes key concepts such as MVC architecture, service-layer abstraction, and HTTP redirection. This application can be extended in the future with features like user accounts, analytics (click tracking), expiration times, or custom aliases.

This project not only strengthens backend development skills using Spring Boot but also introduces how to build, test, and deploy a microservice-like architecture with modern tools.