Simplify

A URL shortener system.

Simplify

System Design Document

Version 1.0

01/09/2025

Table of Contents

Introduction

The System Design Document (SDD) describes how the functional and nonfunctional requirements recorded in the Requirements Document, the preliminary user-oriented functional design.

The SDD describes design goals and considerations, provides a high-level overview of the system architecture, and describes the data design associated with the system, as well as the human-machine interface and operational scenarios.

1.1 Purpose

Simplify -The URL Shortener system is a project to shorten the lengthy URLs.

Lengthy URLs can be a problem to copy,track ,save ,access and hence a short simple and conscise URL is a great alternative.

Simplify does the job in a click and saves the user from a lot of issues.

2. General Overview and Design Guidelines/Approach

This section describes the principles and strategies to be used as guidelines when designing and implementing the system.

2.1 General Overview

The user needs to paste the lengthy URL in the box and click the simplify button.

This will generate the short URL which the user can use it.

The user can also track the count of the visit of particular URL by clicking on the Visit Count button.

2.2 Assumptions/Constraints/Risks

2.2.1 Assumptions

The assumption is that – all the URLs pasted should start from ‘https’.

Have added a notification message stating the same in the application.

2.2.2 Risks

Tracking of malicious URLs is difficult.

3. Design Considerations

3.1 Goals and Guidelines

The Goal Of the project is to generate a short URL as an alternative to a lengthy URL

The URL should start with https Example: https://www.google.com

3.2 Development Methods & Contingencies

Front End

React is used for the front-end and necessary request calls are made to the backend using axios module in react.

Database

Mongo DB is used as the database where the Original URL links are stored with their respective conscise url in documents.

Back End

Node Js is used as the backend server to create the API endpoints and to create and update the records in mongo db.

3.3 Architectural Strategies

MVC architecture is followed in the application.

The Model consists of the Database schema and the controller consists of the API endpoints code.

The routes file consisits of the API callouts on a particular route.

4. System Design

4.1 Business Requirements

For the Database part, there are three parts:

-orginal URL : Original URL pasted by the user

-short URL : Short URL generated by the application

-visit count : Contains the count of the visit of a particular URL.

4.2 Database Design

```
{
{
    shortId: {
      type: String,
      required: true,
      unique: true,
    },
    redirectURL: {
      type: String,
      required: true,
    },
    visitHistory: [{ timestamp: { type: Number } }],
  },
  { timestamps: true }

}
```

5. References

1. https://bitly.com/
2. https://www.shorturl.at/