Url-Shortner



Software Requirement Specification Report

Submitted By Srishti 121115156

15.04.2024

Submitted To

Dr. Balwinder Kaur Dhaliwal Prepared for

Continuous Assessment 3

Spring 2024

School of computer Science and Engineering

1. | P a g e

Software Requirement Specification

1. **Introduction**

The ever-expanding digital landscape has led to an explosion of information, often presented in the form of URLs (Uniform Resource Locators). These URLs, acting as unique identifiers for webpages and resources, can become cumbersomely long and unwieldy. Sharing lengthy URLs can be inconvenient, impacting readability and potentially exceeding character limitations on various platforms. Social media posts, text messages, and even email signatures might not accommodate excessively long URLs.

This Software Requirement Specification (SRS) document presents a solution to this common challenge: a URL shortener application. This innovative tool will bridge the gap between lengthy, cumbersome URLs and a more manageable, user-friendly experience.

The purpose of this SRS document is to clearly define the functionalities, features, and operational requirements of the URL shortener application. By outlining these aspects in detail, we can ensure a successful development process that delivers a robust and valuable solution. This document serves as a roadmap for developers, guiding them in building a URL shortener that effectively addresses the limitations of long URLs and empowers users to share web resources effortlessly.

This introduction offers a more elaborate explanation by:

* + **Expounding on the problem:** Highlighting the increasing prevalence of long URLs due to the vast amount of information online.
  + **Emphasizing the impact:** Detailing the inconveniences caused by long URLs, including sharing difficulties, readability limitations, and character restrictions.
  + **Introducing the solution:** Presenting the URL shortener application as a tool that directly addresses these issues.
  + **Reinforcing the document's purpose:** Clearly stating that the SRS outlines the functionalities and requirements for development, guiding the creation of a valuable solution.

1. | P a g e

Software Requirement Specification

# Purpose

The URL shortener application serves the following key purposes:

* **Enhanced User Experience:** By generating concise and user-friendly alternatives to lengthy URLs, the application aims to improve the overall user experience. Sharing links becomes more convenient, promoting readability and eliminating character limitations encountered on various platforms.
* **Increased Efficiency:** The application streamlines the process of sharing web resources. Users can quickly generate shortened URLs, saving time and effort compared to manually pasting or typing lengthy originals.
* **Improved Brand Image:** Businesses and organizations can leverage shortened URLs to maintain a professional and polished online presence. Shortened URLs often appear cleaner and more aesthetically pleasing, potentially enhancing brand image.
* **Data Analytics Potential:** The application can be designed to capture data associated with shortened URLs. This data, if anonymized and analyzed responsibly, can provide valuable insights into user behavior and content popularity.

# Scope

This section defines the boundaries of the URL shortener application, outlining the functionalities that will be included in the initial development phase.

## In-Scope:

* **Core Functionality:** The application's primary function is to generate shortened URLs from user-provided long URLs. This core functionality should be robust and reliable.
* **Basic Customization:** Users may be allowed to customize shortened URLs to a limited extent. This could involve specifying a preferred ending for the shortened URL (within reason) to enhance memorability or branding.

1. | P a g e

Software Requirement Specification

* **Link Redirection:** When a user clicks on a shortened URL, they should be seamlessly redirected to the original long URL.
* **Basic Analytics: The** application may capture basic data on shortened URL usage, such as the number of clicks. This data should be anonymized and not contain any personally identifiable information (PII).

## Out-of-Scope:

* **Advanced URL Management:** Features like editing or deleting shortened URLs after creation might be excluded from the initial scope.
* **User Accounts and Logins:** The application may not require user accounts for basic URL shortening functionality in the initial phase.
* **Advanced Analytics and Reporting:** Features like detailed click-through rate analysis or geographical tracking might be considered for future development.
* **Integration with Third-Party Services:** Integration with social media platforms or other external services might be explored in later versions.

## Future Considerations:

This section acknowledges potential areas for future development based on user feedback and evolving needs. It allows for flexibility and potential expansion of the application's functionalities.

By clearly defining the scope, we ensure that the development team focuses on the core functionalities of the URL shortener while acknowledging potential areas for future enhancements. This promotes efficient development and avoids feature creep.

1. | P a g e

Software Requirement Specification

# Definitions, Acronyms and Abbreviations

* **URL (Uniform Resource Locator):** A unique identifier used to locate webpages and resources on the internet.
* **Shortened URL:** A concise and user-friendly alternative to a lengthy URL, generated by the application.
* **Long URL:** The original, unabbreviated URL provided by the user.
* **Redirection:** The process of directing a user from a shortened URL to the corresponding long URL.
* **Click-Through Rate (CTR) :** A metric that measures the number of clicks on a shortened URL divided by the number of times it is displayed (optional functionality).
* **API (Application Programming Interface):** A set of protocols and tools for building software applications that enables interaction with the URL shortener (potential future functionality).

# Overview

This section provides a high-level introduction to the URL shortener application, outlining its functionalities and target audience.

* **General Description:** Briefly describe the application as a tool that generates shortened URLs from user-provided long URLs.
* **Target Users:** Identify the intended users of the application, such as individuals sharing links on social media, businesses managing online presence, or content creators including links in their work.
* **Benefits:** Summarize the key benefits of using the URL shortener, such as improved user experience, increased efficiency, enhanced brand image, and potential data analytics insights.

1. | P a g e

Software Requirement Specification

* **High-Level Functionalities:** Provide a concise overview of the application's core functionalities. This could include URL shortening, basic customization options , link redirection, and basic analytics tracking.
* **System Architecture:** Briefly describe the overall architecture of the application, outlining the main components involved in URL shortening and redirection.

# General Description

The URL shortener application is a software tool designed to address the challenge of cumbersome and lengthy URLs. These long URLs can be difficult to share, hindering readability and potentially exceeding character limitations on various platforms.

This application bridges this gap by providing a user-friendly solution. Users can input long URLs, and the application will generate concise, shortened alternatives. These shortened URLs are easier to share, manage, and integrate into various communication channels.

The application aims to:

* + **Enhance User Experience:** Sharing links becomes more convenient and aesthetically pleasing with shortened URLs.
  + **Increase Efficiency:** Users can quickly generate shortened URLs, saving time and effort compared to manually pasting or typing lengthy originals.
  + **Improve Brand Image:** Businesses and organizations can leverage shortened URLs to maintain a professional and polished online presence.

This general description provides a high-level overview of the URL shortener application's purpose and functionality. It highlights the challenges addressed and the benefits offered to users.

1. | P a g e

Software Requirement Specification

# Product Perspective

The URL shortener application is viewed from the following product perspectives:

* + - **End User:** The primary user of the application is anyone who needs to share web resources online. This includes individuals sharing links on social media platforms, bloggers including links in their content, businesses managing their online presence through various channels, and anyone who encounters lengthy URLs they want to make more manageable.
      * The user expects a simple and intuitive interface for entering long URLs and generating shortened alternatives.
      * The user desires shortened URLs that are concise, easy to remember, and aesthetically pleasing.
      * The user wants the application to be reliable and ensure seamless redirection to the original webpage when clicking on a shortened URL.
      * Depending on the implementation, the user might appreciate basic customization options for shortened URLs and the ability to track basic analytics on shortened URL usage.
    - **System Administrator:** If the application is deployed for internal use within an organization, a system administrator might be responsible for managing user access, monitoring system health, and potentially configuring advanced settings.
      * The system administrator requires a user management system to control user access and permissions.
      * The administrator needs tools to monitor system performance and ensure smooth operation.
      * Depending on the application's complexity, the administrator might require configuration options for features like customization rules or analytics reporting.
    - **Developer:** The developers building and maintaining the URL shortener application need clear and concise technical specifications outlined in this SRS document.
      * This document serves as a roadmap for developers, providing details on functionalities, features, and operational requirements.

1. | P a g e

Software Requirement Specification

* + - * The defined scope ensures developers focus on core functionalities while acknowledging potential areas for future enhancements.

By outlining these product perspectives, the SRS document caters to the needs of various stakeholders involved in the development, use, and maintenance of the URL shortener application. This ensures a well-rounded understanding of the product from different viewpoints.

# Product Functions

This section details the core functionalities of the URL shortener application:

* + - **URL Shortening:** This is the primary function of the application. Users should be able to enter a long URL into a designated field. The application will then process the URL and generate a shortened alternative.
      * **Supported URL Formats:** The application should specify the types of URLs it can handle (e.g., HTTP, HTTPS).
      * **Shortened URL Length:** Define the target length for shortened URLs (e.g., 6 characters, 8 characters).
      * **Collision Handling:** Describe the strategy for handling situations where a generated shortened URL already exists for a different long URL. This could involve appending random characters or offering alternative suggestions.
    - **Link Redirection:** When a user clicks on a shortened URL, the application should seamlessly redirect them to the corresponding long URL.
      * **Redirection Method:** Specify the redirection method used (e.g., HTTP 301 redirect, JavaScript redirect).
      * **Error Handling:** Outline the behavior in case the original long URL becomes unavailable. This could involve displaying an error message or redirecting to a custom error page.
    - **Basic Customization:** The application may offer limited customization options for shortened URLs.

1. | P a g e

Software Requirement Specification

* + - * **Customizable Ending:** Users might be able to specify a preferred ending for the shortened URL within predefined parameters. This could enhance memorability or align with branding.
    - **Basic Analytics Tracking :** The application can optionally capture basic data on shortened URL usage.
      * **Anonymized Data:** This data should be anonymized and not contain any personally identifiable information (PII).
      * **Metrics:** The application might track metrics like the number of clicks on a shortened URL. This data can provide insights into user behavior and content popularity (if implemented responsibly).

# General Constraints

The URL shortener application needs to operate within certain limitations to ensure functionality, security, and scalability. Here are some key general constraints to consider:

* + - **Supported Characters in Shortened URLs:** Define the allowed character set for shortened URLs. This might include alphanumeric characters (a-z, A-Z, 0-9) and special characters (e.g., -, \_) with limitations to avoid confusion or potential security risks.
    - **Shortened URL Length:** Specify the target length for shortened URLs. This should balance readability and functionality. Longer URLs might be easier to remember but offer less reduction in length.
    - **URL Shortening Algorithm:** Describe the chosen algorithm for generating shortened URLs. It should be efficient, avoid predictable patterns, and minimize collisions (generating the same shortened URL for different long URLs).
    - **Maximum URL Length:** Define the maximum length of a long URL the application can handle. This ensures the application can process most

1. | P a g e

Software Requirement Specification

common URLs while avoiding potential performance issues with excessively long ones.

* + - **Rate Limiting:** Consider implementing rate limiting to prevent abuse or malicious use of the application. This could involve restricting the number of shortened URLs a user can generate within a specific timeframe.
    - **Security:** The application should be designed with security in mind. This includes measures to prevent unauthorized access, manipulation of shortened URLs, or malicious redirection.

These constraints ensure the URL shortener application functions effectively within a defined framework. They address potential issues like character limitations, URL length management, and security considerations.

# Assumptions and dependencies

This section outlines the assumptions and dependencies for the URL shortener application:

## Assumptions:

* + - Users have a basic understanding of the internet and URLs.
    - Users have a reliable internet connection to access and use the application.
    - The long URLs provided by users are valid and publicly accessible webpages.
    - Users will respect the terms of service governing the use of the application.

## Dependencies:

* + - **External Services :** If the application relies on external services for functionalities like URL shortening or data storage, their availability and reliability are crucial.
    - **Underlying Technologies:** The application depends on various underlying technologies to function, such as web servers, databases , and programming languages. The successful operation of the application hinges on the proper functioning of these technologies.
    - **Operating System:** The application might have dependencies on specific operating systems or platforms for deployment.

1. | P a g e

Software Requirement Specification

By outlining these assumptions and dependencies, we acknowledge external factors that can influence the development and operation of the URL shortener application. This helps to manage expectations and identify potential risks during the development process.

# Specific Requirements

## External Interface Requirements

The URL shortener application interacts with various external entities. This section defines the requirements for these interfaces:

## User Interface (UI):

* + The UI should be user-friendly and intuitive, allowing users to easily enter long URLs and view generated shortened URLs.
  + The UI should provide clear instructions and feedback to users throughout the URL shortening process.
  + The UI may offer optional features like basic customization options for shortened URLs .
  + The UI should display error messages or appropriate feedback in case of issues like invalid URLs or exceeding usage limits.

## Web Interface:

* + If the application is accessible through a web interface, it should follow web development best practices for responsiveness and accessibility.
  + The web interface should be compatible with major web browsers on various devices (desktop, mobile, tablets).
  + The web interface should implement security measures to protect user data and prevent unauthorized access.

## API :

* + If the application offers an API for programmatic access to its functionalities, the API should be well-documented and easy to integrate with other applications.

1. | P a g e

Software Requirement Specification

* + The API should provide clear endpoints for functions like URL shortening, retrieving shortened URL information, and potentially accessing basic analytics data.
  + The API should implement authentication and authorization mechanisms to control access and prevent misuse.

## External Services:

* + If the application relies on external services for URL shortening, data storage, or other functionalities, the interfaces with these services should be clearly defined and documented.
  + The application should handle potential service outages or errors gracefully, minimizing disruption to users.

This section focuses on the interfaces between the URL shortener application and the external world, including users, web browsers, and potential APIs or external services. Defining these requirements ensures a smooth user experience, efficient integration with other systems, and robust operation of the application.

* 1. **Functional Requirements**

This section details the specific functionalities that the URL shortener application must provide:

## FR-1: URL Shortening

* + - The application shall provide a user interface for entering a long URL.
    - The application shall validate the entered URL format (e.g., HTTP, HTTPS). [FR-1.1]
    - Upon valid URL submission, the application shall generate a shortened URL. [FR-1.2]
    - The shortened URL length shall be within the defined target range (e.g., 6-8 characters). [FR-1.3]
    - The application shall handle URL shortening algorithm collisions (duplicate shortened URLs) using a defined strategy (e.g., appending random characters). [FR-1.4]

1. | P a g e

Software Requirement Specification

## FR-2: Link Redirection

* + - When a user clicks on a shortened URL, the application shall redirect them to the corresponding long URL. [FR-2.1]
    - The application shall utilize a defined redirection method (e.g., HTTP 301 redirect). [FR-2.2]
    - In case the original long URL becomes unavailable, the application shall display a user-friendly error message or redirect to a custom error page. [FR- 2.3]

## FR-3: Basic Customization

* + - The application may offer an option for users to specify a preferred ending for the shortened URL within predefined parameters. [FR-3.1]
    - The allowed characters for custom endings should be clearly defined to avoid confusion or security risks. [FR-3.2]

## FR-4: Basic Analytics Tracking

* + - The application may optionally capture anonymized data on shortened URL usage. [FR-4.1]
    - This data should not contain any personally identifiable information (PII). [FR- 4.2]
    - The application may track metrics like the number of clicks on a shortened URL. [FR-4.3]

## Additional Functional Requirements :

* + - FR-x: Implement rate limiting to restrict the number of shortened URLs a user can generate within a specific timeframe.
    - FR-y: Integrate with social media platforms to allow users to directly share shortened URLs.

This section outlines the core functionalities of the URL shortener application, along with potential optional features. Each functional requirement (FR) is broken down into specific details for clarity and to guide development.

1. | P a g e

Software Requirement Specification

# Non-Functional Requirements

This section details the non-functional requirements for the URL shortener application, focusing on performance, usability, reliability, and security. These requirements define how the application should behave beyond its core functionalities.

## Performance:

* + - **Response Time:** The application should generate shortened URLs and perform redirects within an acceptable timeframe (e.g., less than 2 seconds). [NFR-1.1]
    - **Scalability:** The application should be scalable to handle an increasing number of users and shortened URLs without significant performance degradation. [NFR-1.2]

## Usability:

* + - **User Interface:** The user interface should be intuitive and user-friendly, allowing users to easily navigate and interact with the application. [NFR-2.1]
    - **Error Handling:** The application should provide clear and informative error messages to guide users in case of invalid URLs or other issues. [NFR-2.2]
    - **Accessibility :** The application should adhere to accessibility guidelines to ensure usability for users with disabilities.

## Reliability:

* + - **Availability:** The application should be highly available with minimal downtime to ensure users can access its functionalities. [NFR-3.1]
    - **Uptime Target:** Define a target uptime percentage (e.g., 99.5%) to represent the expected availability of the application. [NFR-3.2]
    - **Data Integrity:** The application should ensure the accuracy and consistency of stored data, such as shortened URLs and their corresponding long URLs. [NFR-3.3]

## Security:

* + - **Input Validation:** The application should validate user input (URLs) to prevent malicious code injection or other security vulnerabilities. [NFR-4.1]

1. | P a g e

Software Requirement Specification

* + - **Data Security:** Any user data collected by the application (e.g., anonymized usage data) should be stored securely and protected from unauthorized access. [NFR-4.2]
    - **Secure Redirection:** The application should utilize secure redirection methods (e.g., HTTPS) to protect user data during redirection. [NFR-4.3]

## Additional Non-Functional Requirements :

* + - NFR-x: Implement logging mechanisms to track application activity and identify potential issues.
    - NFR-y: Define disaster recovery procedures to ensure quick recovery in case of system outages.

These non-functional requirements ensure the URL shortener application performs efficiently, offers a user-friendly experience, operates reliably, and maintains a secure environment.

# Design Constraints

Building a URL shortener application requires careful consideration of design constraints to ensure functionality, scalability, and user experience. Here are some key design constraints to address:

* + - **Shortened URL Length:** There's a trade-off between shortened URL length and uniqueness. Shorter URLs are more user-friendly but increase the risk of collisions (generating the same shortened URL for different long URLs). Define a target length (e.g., 6-8 characters) that balances readability with collision avoidance.
    - **Character Set:** Restrict the allowed characters in shortened URLs to avoid ambiguity and potential security risks. Consider alphanumeric characters (a-z, A-Z, 0-9) and special characters (e.g., -, \_) with limitations (avoid characters like "/" or "?" that might be misinterpreted).
    - **Collision Handling:** Develop a strategy to handle URL shortening collisions. This could involve:

1. | P a g e

Software Requirement Specification

* + - * Appending random characters to the shortened URL.
      * Offering alternative suggestions to users if a collision occurs.
      * Implementing a more complex algorithm to minimize collisions (consider trade-offs between performance and effectiveness).
    - **Maximum Long URL Length:** Define a limit on the maximum length of a long URL the application can handle. This ensures processing most common URLs while avoiding potential performance issues with excessively long ones.
    - **Database Selection :** If the application stores shortened URLs and their mappings to long URLs, choose a database technology that offers efficient storage, retrieval, and scalability based on anticipated usage volume.
    - **Rate Limiting :** Consider implementing rate limiting to prevent abuse or malicious use of the application. This could involve restricting the number of shortened URLs a user can generate within a specific timeframe.
    - **API Design :** If the application offers an API for programmatic access, design a well-documented and intuitive API with clear endpoints, authentication mechanisms, and error handling to facilitate integration with other systems.

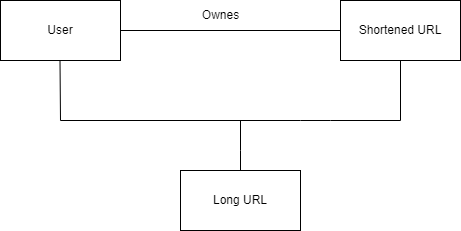
These design constraints provide a framework for developing a URL shortener application that is efficient, user-friendly, and scalable.

**4. ER Model of URL Shortener Generator**

An Entity-Relationship Diagram (ERD) for a URL Shortener Generator models the entities and their relationships within the system. Below is a simplified ERD for a URL Shortener Generator. In Synopsys we make a rough ER Diagram to give a idea about the working of the project*.*

1. | P a g e

Software Requirement Specification



# Analysis Models

## Context Diagram

**Title:** URL Shortener Application Context Diagram

## Content:

* + A single external entity named "User" interacts with the URL shortener application.
  + The user provides a long URL as input.
  + The application generates a shortened URL and returns it to the user.

The context diagram offers a high-level overview of the URL shortener application's interaction with the external world. It depicts the user as the sole entity exchanging information with the application. The user submits a long URL, and the application responds with a shortened URL.

## Level 0 DFD

**Title:** URL Shortener Application Level 0 DFD

## Content:

* + The level 0 DFD expands on the context diagram, revealing the internal processes of the URL shortener application.

1. | P a g e

Software Requirement Specification

* + A single process, named "Shorten URL," represents the core functionality of the application.
  + The user provides a long URL as input to this process.
  + The process interacts with two data stores:
    - A "Long URL Database" stores the original long URLs.
    - A "Shortened URL Database" stores the shortened URLs and their corresponding long URLs.
  + The shortened URL generated by the process is returned to the user.

The level 0 DFD provides a more detailed view of the URL shortener application. It introduces two data stores: one for long URLs and another for shortened URLs with mappings to their corresponding long URLs. The "Shorten URL" process interacts with these data stores to fulfill its core function of generating shortened URLs.

## Level 1 DFD

**Title:** URL Shortener Application Level 1 DFD

## Content:

* + The level 1 DFD further decomposes the "Shorten URL" process from the level 0 DFD.
  + It can be broken down into sub-processes such as:
    - "Validate URL": Ensures the entered URL is formatted correctly (e.g., HTTP, HTTPS).
    - "Generate Shortened URL": Creates a unique shortened URL using an algorithm.
    - "Check for Collisions": Verifies if the generated shortened URL already exists in the database.
    - "Store Long URL" : Stores the original long URL in the database if not already present.
    - "Store Mapping": Saves the association between the shortened URL and the corresponding long URL in the database.

The level 1 DFD is optional and provides a more granular view of the URL shortening process. It showcases potential sub-processes involved in validating the URL, generating a shortened URL, handling collisions, and storing necessary data in

1. | P a g e

Software Requirement Specification

the respective databases. This level of detail can be beneficial for complex URL shortener applications with additional features.

1. | P a g e

Software Requirement Specification