URL SHORTENER

PROJECT DOCUMENTATION

---

1. Introduction

- **1.1 Overview**

The URL Shortener is a web application designed to take long URLs and provide shorter, more manageable links. This is useful for sharing links on social media, in emails, or anywhere where character limits or readability are important. This simplified version demonstrates the core functionality of URL shortening.

- **1.2 Purpose**

The primary purpose of this project is to illustrate the basic principles behind URL shortening. It provides a user interface where a user can input a long URL and receive a shortened version.

2. Project Goals

- **2.1 Core Objectives**
    - Provide a user interface for entering a long URL.
    - Generate a shortened URL from the input.
    - Display the shortened URL to the user.
    - Ensure the shortened URL, when clicked, redirects to the original long URL.
- **2.2 Functional Scope**
    - This version focuses on the client-side logic of generating a shortened URL.
    - It *simulates* the shortening process with a random string; it does not connect to a database or external service to store and retrieve URL mappings.

3. Features and Working

- **3.1 Features**
    - **URL Input:** A text field where users can enter the long URL they want to shorten.
    - **URL Shortening:** A button that, when clicked, generates a shortened version of the input URL.
    - **Shortened URL Display:** Displays the generated shortened URL to the user.

    - **Redirection:** The displayed shortened URL is a clickable link that, in this demo, will redirect to the original URL entered by the user.

- **3.2 Working**

1. **Input:** The user enters a long URL into the input field.
2. **Submission:** The user clicks the "Shorten URL" button.
3. **Shortening (Simulation):**
    - The JavaScript code retrieves the long URL from the input field.
    - It generates a random string to represent the "shortened" part of the URL.
    - It constructs a simulated shortened URL (e.g., https://short.ly/randomString).
4. **Display:** The generated shortened URL is displayed on the page as a clickable link.
5. **Redirection (Simulated):** When the user clicks the displayed shortened URL, the browser is directed to the original long URL (this works because the href attribute of the link is set to the original URL).

4. Technology Used

- **4.1 HTML (Hypertext Markup Language)**
    - Provides the structure of the web page.
    - Includes elements for the heading, input field, button, and display area.
- **4.2 CSS (Cascading Style Sheets)**
    - Handles the styling and visual presentation of the page.
    - Defines the layout, colors, fonts, and overall aesthetics.
- **4.3 JavaScript**
    - Implements the logic for:
        - Getting the input URL.
        - Generating the simulated shortened URL.
        - Displaying the result.

5. Implementation Details

- **5.1 HTML Structure**

    - <h1> for the page title ("URL Shortener").
    - <div class="container"> to hold the input and output elements.
    - <input type="text" id="longURL"> for the user to enter the long URL.
    - <button onclick="generateShortURL()"> to trigger the shortening process.
    - <div id="shortenedURL"> to display the generated short URL.
- **5.2 CSS Styling**
    - **Dark Theme:** A dark background with lighter text for contrast and a modern look.
    - **Layout:** Flexbox is used to center the content vertically and horizontally.
    - **Input and Button Styling:** Custom styles for the input field and button to match the theme.
    - **Container:** A styled container (<div>) to group the elements and provide visual separation.
- **5.3 JavaScript Logic**
    - generateShortURL() function:
        - Gets the value from the longURL input field.
        - Validates if the input is empty.
        - Generates a random string using Math.random().toString(36).substring(7). **NOTE:** This is a simplified approach; in a real-world application, a more robust and unique ID generation method would be used, often involving a database.
        - Constructs the simulated short URL.
        - Updates the innerHTML of the shortenedURL div to display the result as a clickable link. The href attribute of the <a> tag is set to the original long URL.

6. Theme

- **6.1 Color Palette**
    - Background: Dark (#101010, #1a1a2e)
    - Text: Light blue/grayish (#aad8ff)
    - Accent: Bright cyan (#00f0ff, #00c8cc on hover)

- **6.2 Typography**
    - Font: Arial, sans-serif
- **6.3 Visual Style**
    - Clean, modern, and minimalist design.
    - Dark theme with vibrant accents for good contrast.
    - Rounded corners for a softer look.
    - Subtle box shadow for depth.

7. Code Snippets

- **7.1 HTML Input and Button**

HTML

```html
<input type="text" id="longURL" placeholder="Enter your long URL here..." />
<button onclick="generateShortURL()">Shorten URL</button>
```

- **7.2 JavaScript Function**

JavaScript

```javascript
function generateShortURL() {
  const longURL = document.getElementById('longURL').value;
  if (longURL === '') {
    alert('Please enter a URL.');
    return;
  }

  // Generate a random string as the shortened URL (for demo purposes)
  const randomString = Math.random().toString(36).substring(7);

  // Simulate the shortened URL
  const shortURL = `https://short.ly/${randomString}`;

  // Display the shortened URL
```

```
   document.getElementById('shortenedURL').innerHTML = `Shortened URL: <a href="<span class="math-inline">\{longURL\}" target\="\_blank"\></span>{shortURL}</a>`;
}
```

- **7.3 CSS Styling**

CSS

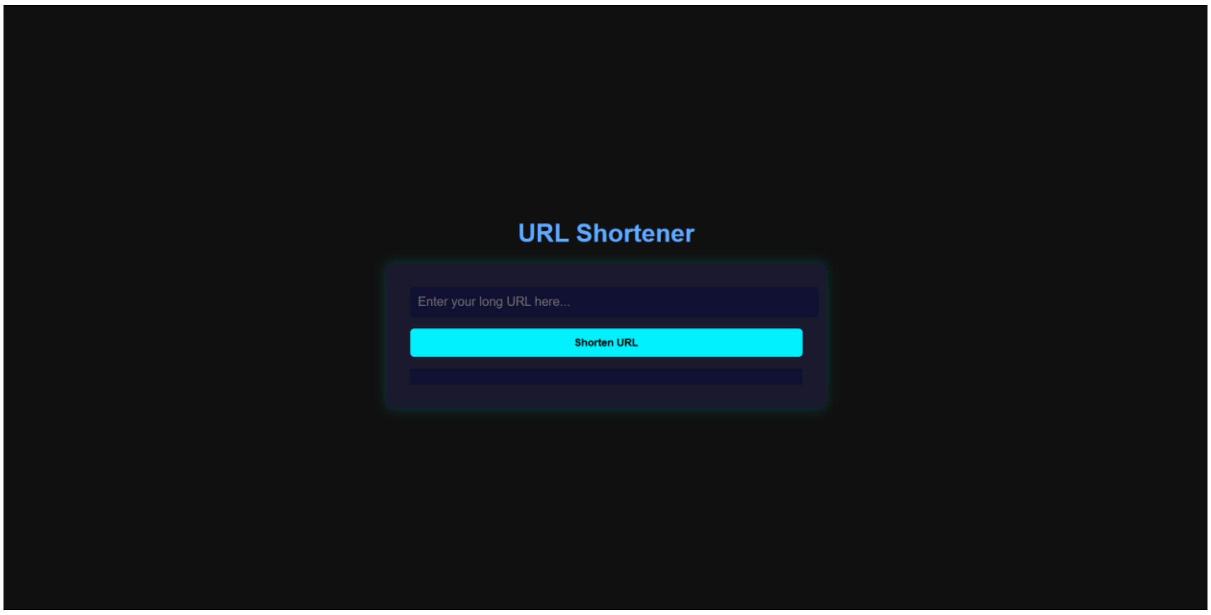
```
body {
   background: #101010;
   color: #aad8ff;
   font-family: Arial, sans-serif;
   display: flex;
   flex-direction: column;
   align-items: center;
   justify-content: center;
   height: 100vh;
   margin: 0;
}

input {
   width: 100%;
   padding: 10px;
   margin-bottom: 15px;
   border: none;
   border-radius: 5px;
   background-color: #101030;
   color: #aad8ff;
   font-size: 16px;
}
```

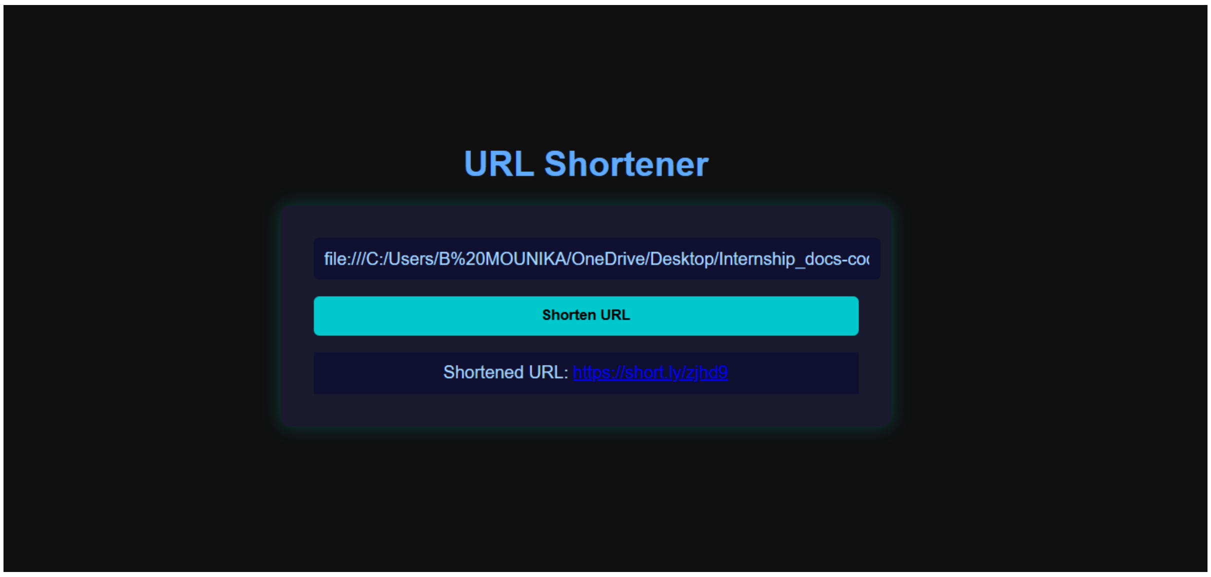
```
button {
    padding: 10px 20px;
    background: #00f0ff;
    color: #000;
    border: none;
    border-radius: 5px;
    font-weight: bold;
    cursor: pointer;
    width: 100%;
    transition: background 0.3s ease;
}
```

8. Output Photos

- **8.1 Initial State**
    - The page displays the title "URL Shortener."
    - There's an input field labeled "Enter your long URL here...".
    - A button labeled "Shorten URL" is below the input.
    - The area where the shortened URL will be displayed is empty.

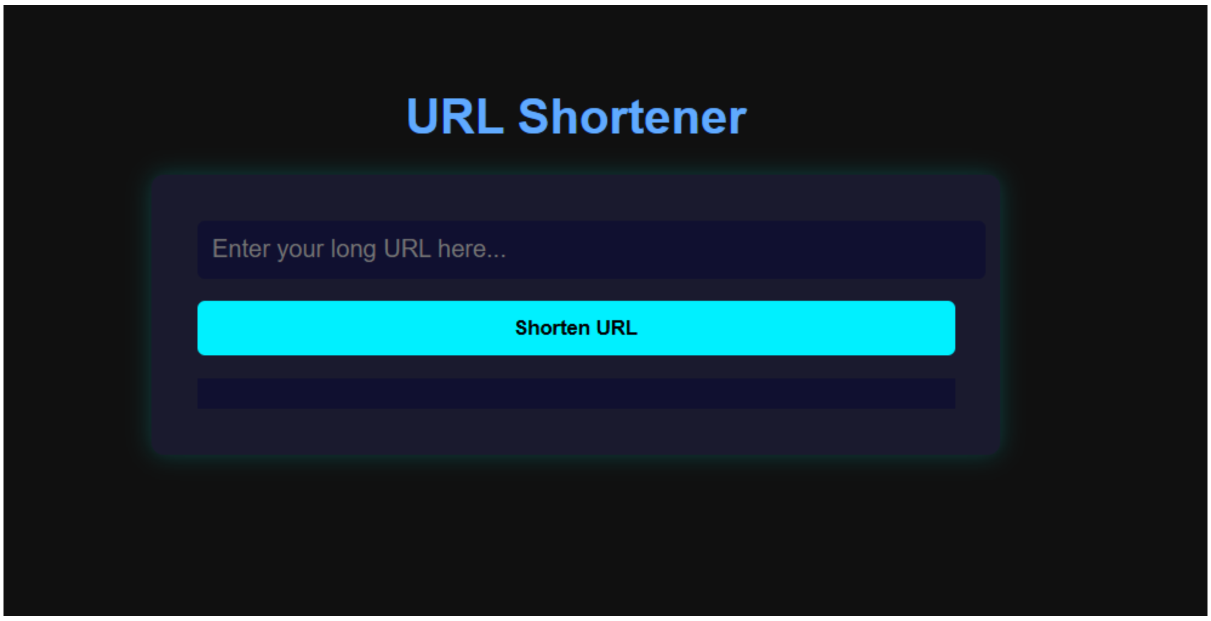


- **8.2 After Entering a URL and Clicking "Shorten URL"**
  - The shortened URL is displayed below the button.
  - The shortened URL is a clickable link.
  - The link, if clicked, would redirect to the originally entered URL.



file:///C:/Users/B%20MOUNIKA/OneDrive/Desktop/Internship_docs-co

Shortened URL: https://short.ly/zjhd9



9. Limitations and Feature Enhancements

- **9.1 Limitations**
  - **Simulated Shortening:** The URL shortening is simulated using a random string. It does not actually store URL mappings or generate truly unique, short codes.

    - **No Persistence:** The shortened URLs are not stored. If you refresh the page, the generated URL is lost.
    - **Basic Validation:** The input validation is very basic (just checking for empty input). It doesn't validate the format of the URL.
    - **No Custom Short Codes:** Users cannot specify their own custom short codes.
    - **No Analytics:** There's no tracking of clicks or usage statistics for the shortened URLs.
- **9.2 Feature Enhancements**
    - **Database Integration:** Use a database (e.g., MySQL, PostgreSQL, MongoDB) to store the mappings between long and short URLs. This is crucial for persistence and proper functionality.
    - **Unique Short Code Generation:** Implement a robust algorithm to generate unique and short codes (e.g., using a combination of hashing and base62 encoding).
    - **URL Validation:** Add more comprehensive validation to ensure the user enters a correctly formatted URL.
    - **Custom Short Codes:** Allow users to optionally specify a custom short code (if available).
    - **Redirection Tracking:** Track the number of times a shortened URL is clicked.
    - **User Authentication:** Add user accounts to manage and track shortened URLs.
    - **API Endpoint:** Create an API endpoint to allow other applications to use the URL shortening service.
    - **Link Expiration:** Implement the ability to set expiration dates for shortened URLs.
    - **Analytics Dashboard:** Provide a dashboard to view statistics on URL usage.

10. Conclusion

This URL Shortener project provides a fundamental demonstration of the concept of URL shortening. While it uses a simplified approach for generating short URLs, it effectively illustrates the user interface and basic workflow involved. The project highlights the use of HTML, CSS, and JavaScript to create an interactive web application. To create a production-ready URL shortening service, the limitations outlined above would need to be addressed with more advanced techniques and technologies.