

NAME :
SARIYA FATIMA

**TOPIC: URL SHORTNER
WITH QR CODE And
VISUALIZATION
FEATURES**

ABSTRACT

The **URL Shortener with QR Code and Visualization Features** application offers a comprehensive solution for transforming lengthy URLs into concise, shareable links while generating QR codes for easy access. This platform enables users to track link performance through click monitoring and provides robust data visualization tools, allowing for insightful analysis of user engagement. With an intuitive interface, users can manage their links effortlessly, creating custom short links for branding and ensuring secure access through HTTPS protocols. Designed for both individual and business use, the application enhances online presence and maximizes engagement by delivering real-time analytics and seamless integration with third-party services.

KEY FEAYURES:

1. **URL Shortening:** Convert long URLs into short, shareable links.
2. **QR Code Generation:** Create scannable QR codes for each shortened link.
3. **Click Tracking:** Monitor the number of clicks on each link.
4. **Data Visualization:** Display click data through interactive charts and graphs.
5. **User-Friendly Interface:** Easy-to-navigate design for seamless user experience.
6. **Link Management:** View and delete shortened links from a dashboard.
7. **Real-Time Analytics:** Access up-to-date metrics on link performance.
8. **Responsive Design:** Optimized for use on mobile devices and tablets.
9. **Secure Links:** Ensure data security with HTTPS and encryption.

CHAPTER 1

INTRODUCTION

In the digital age, managing web addresses (URLs) effectively has become an important aspect of internet usage, especially with the rise of online marketing, social media, and digital communications. Long URLs, such as those generated by websites with extensive query parameters, are often cumbersome to share and use. They may break in emails, be difficult to remember, or simply look unappealing in messages or social media posts. This leads to the increasing demand for URL shortening services, which can transform long URLs into concise, shareable links.

URL Shortening is the process of taking a lengthy URL and creating a new, shorter one that redirects to the original. These shortened URLs are easier to share and manage but offer minimal additional features. The basic functionality of shortening URLs solves the problem of unwieldy links, but users today expect more advanced functionality, such as tracking link performance, generating QR codes for quick scanning, and visualizing the data for better insights. This is where the **URL Shortener with QR Code and Visualization Features** project comes in.

This project aims to enhance the traditional URL shortener by incorporating **three main features**:

1. **QR Code Generation:** For each shortened URL, a corresponding QR code is generated automatically. This allows users to share links more easily in physical formats (such as printed materials) and scan them quickly with smartphones.
2. **Visit Tracking:** The system tracks the number of visits to each shortened URL, providing users with real-time data on the number of times their links are accessed.
3. **Data Visualization:** Users can view visit statistics in a visual format, such as charts or graphs, making it easy to analyze link performance over time.

The project will use **MongoDB** as its database to store URLs, and tracking data. MongoDB's flexibility and scalability make it ideal for managing a large amount of data efficiently. **Node.js** and **Express.js** will be used for backend development, enabling the server to handle

requests, generate shortened URLs, and store data securely. The frontend, built with **React**, will provide users with a seamless and interactive experience, enabling them to manage their shortened links, view visit statistics, and delete URLs when needed.

Technology Stack

To implement the system effectively, the following technologies are utilized:

- **MongoDB:** A NoSQL database used for storing shortened URLs, and visit data. MongoDB's document-based model is ideal for storing varied data and scaling as user activity grows.
- **Node.js and Express.js:** These technologies handle server-side logic, including URL shortening, QR code generation, and visit tracking. Node.js offers asynchronous processing, making the system capable of handling numerous requests simultaneously.
- **React.js:** The frontend framework used to create an interactive user interface that provides real-time updates and visual feedback for link tracking and management.
- **QR Code Generation Library:** A library that automatically generates QR codes for each shortened URL.
- **Data Visualization Library (e.g., Chart.js or D3.js):** For rendering graphical representations of link activity, such as charts or graphs.

CHAPTER 4

DESIGN

4.1 Architecture:

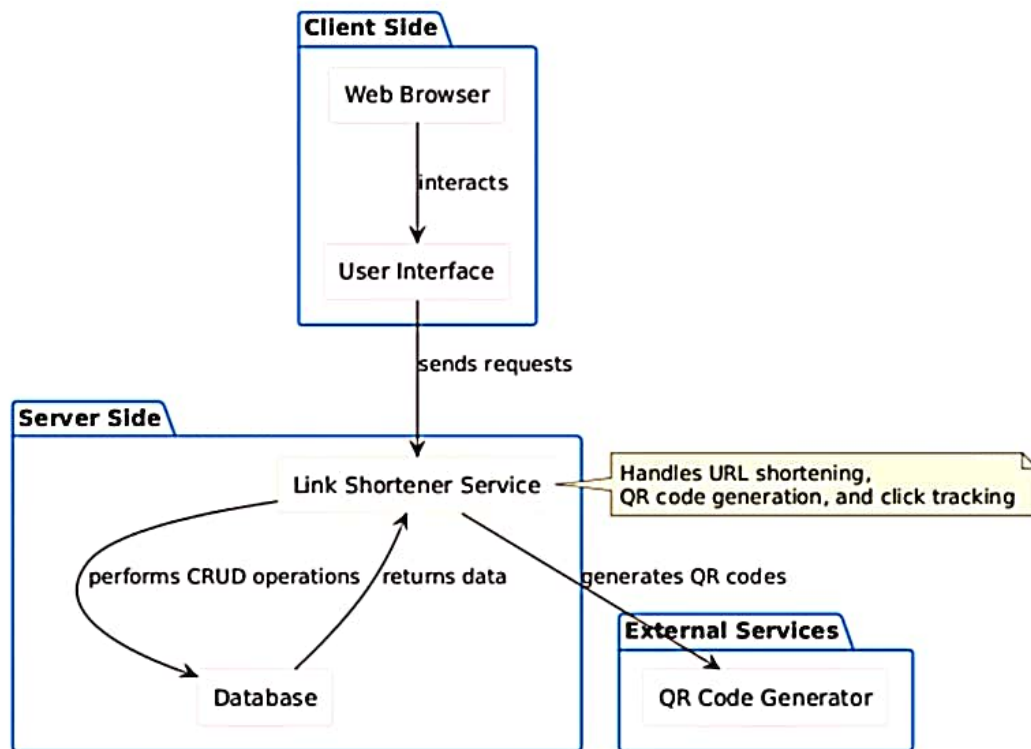


Figure 4.1

4.2 Flow Chart:

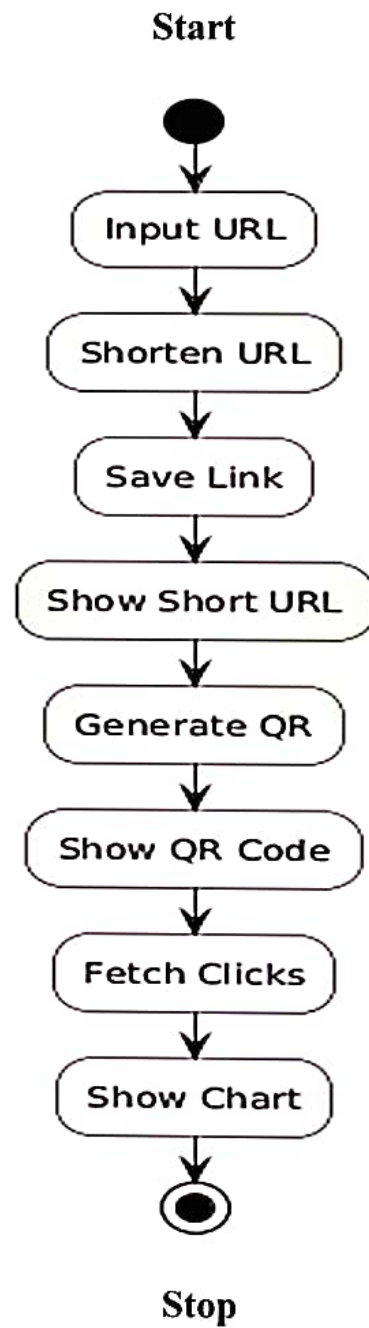


Figure 4.2

4.3 UML Diagrams:

4.3.1 Use Case Diagram:

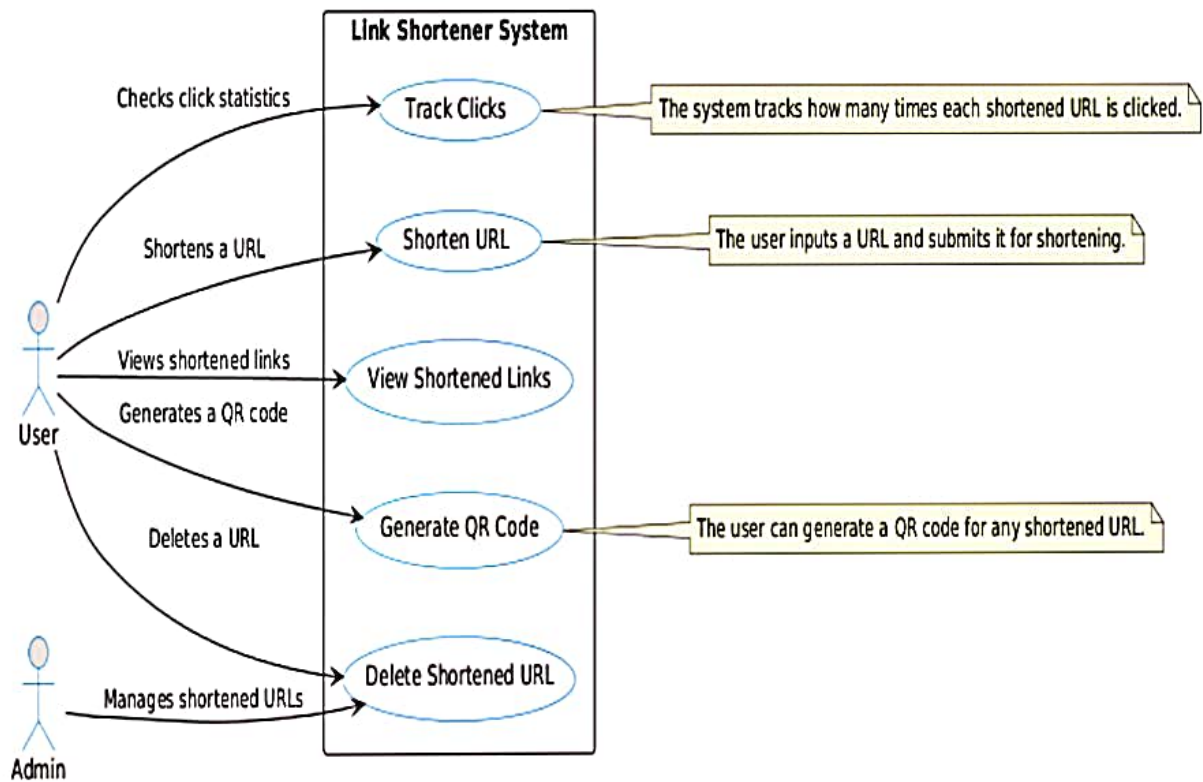


Figure 4.3.1

4.3.2 UML Class Diagram:

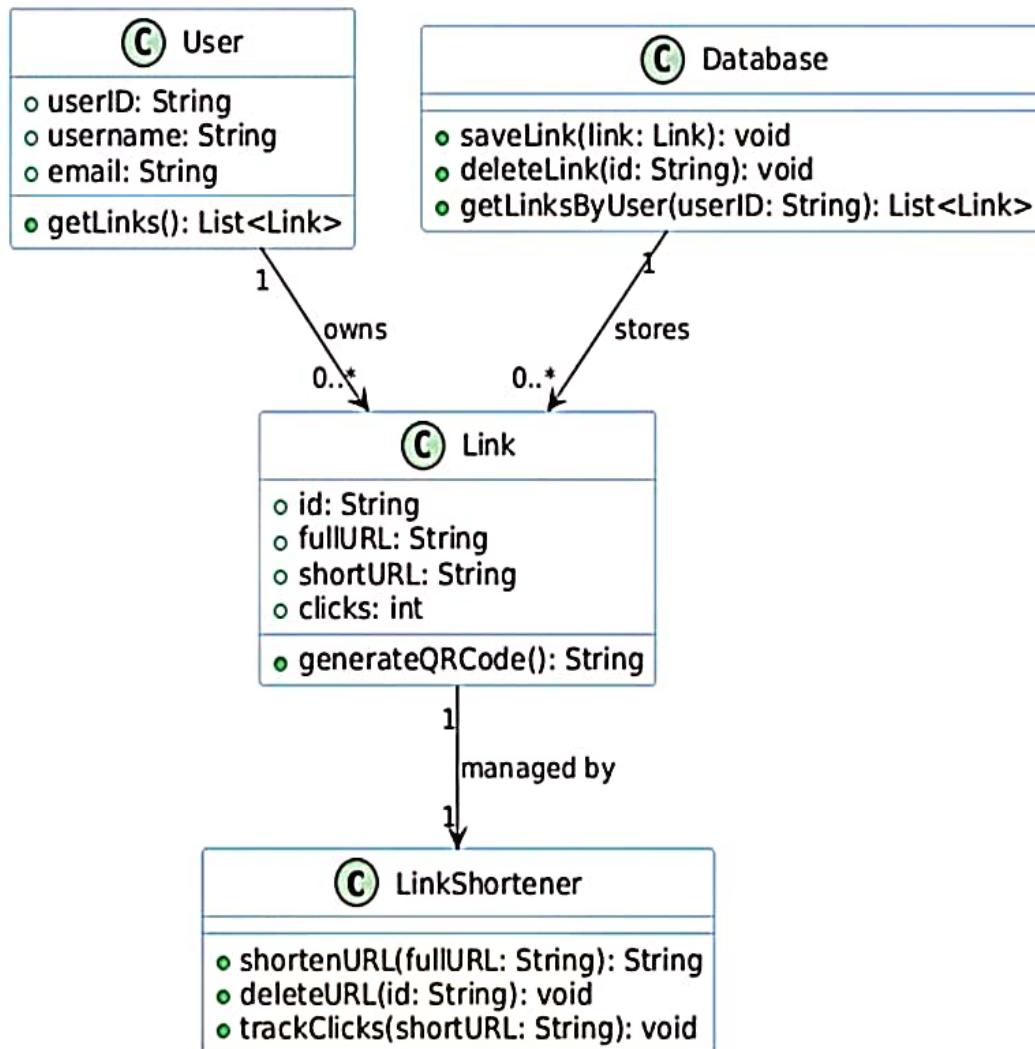


Figure 4.3.2

4.4 Sequence Diagram:

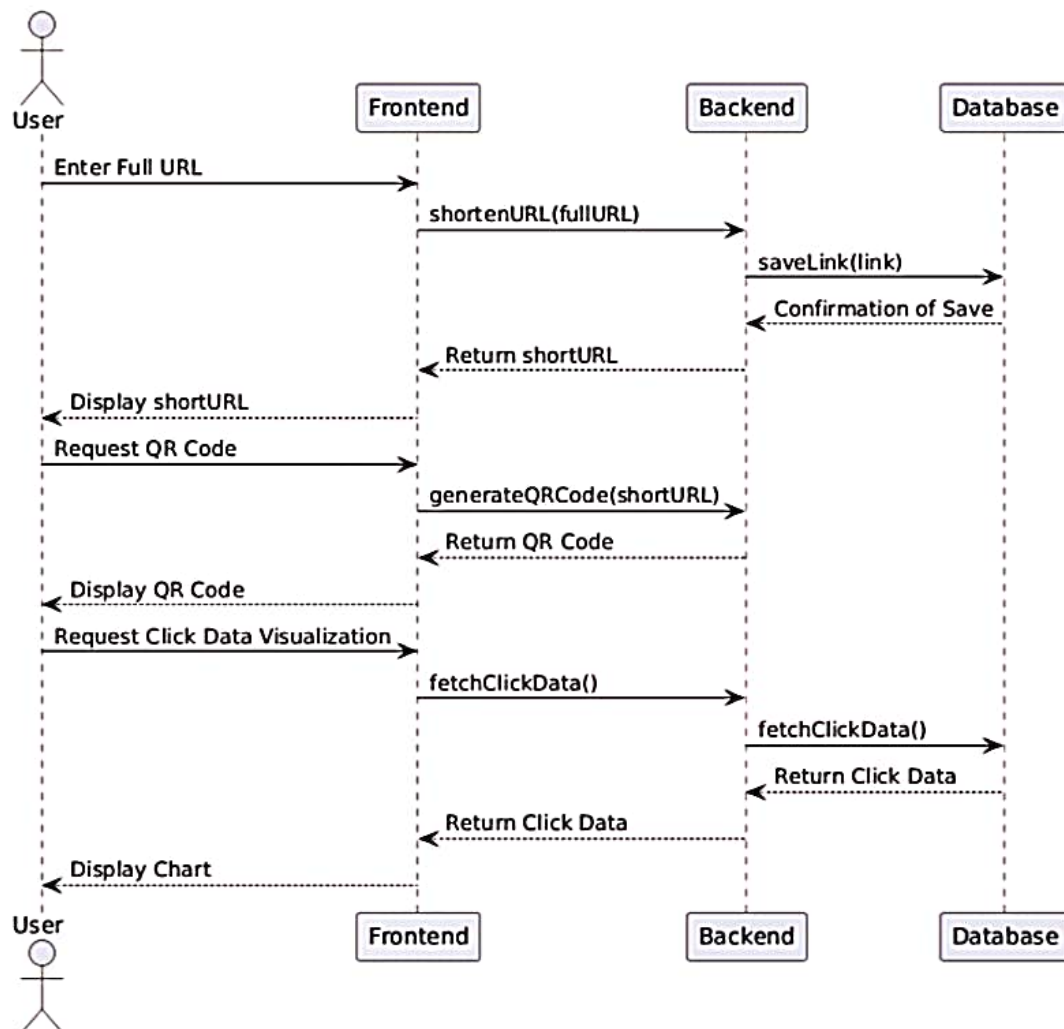


Figure 4.4

CHAPTER 5

IMPLEMENTATION

5.1 Source Code:

5.1.1 Frontend:

Index.ejs:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Link Shortener</title>

  <link href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" rel="stylesheet" />

  <link rel="stylesheet" href="css/style.css">

  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

  <script src="https://cdn.jsdelivr.net/npm/qrcodejs/1.0.0/qrcode.min.js"></script>

  <!--QR Code JS -->

  <script>

    function showAlert() {

      alert('Your link has been submitted successfully!');
```

```

}

// Function to generate QR code and hide the button
function generateQRCode(shortUrl, id) {
    var qrcodeContainer = document.getElementById('qrcode-' + id);
    var qrButton = document.getElementById('qr-btn-' + id);
    qrcodeContainer.innerHTML = ''; // Clear previous QR code
    new QRCode(qrcodeContainer, {
        text: shortUrl,
        width: 80, // QR code size
        height: 80
    });
    qrButton.style.display = 'none'; // Hide the button after generating the QR code
}

// Function to render chart for clicks
function renderClickChart(labels, data) {
    var ctx = document.getElementById('clickChart').getContext('2d');
    new Chart(ctx, {
        type: 'bar',
        data: {
            labels: labels,

```

```

    datasets: [{
        label: '# of Clicks',
        data: data,
        backgroundColor: 'rgba(75, 192, 192, 0.6)',
        borderColor: 'rgba(75, 192, 192, 1)',
        borderWidth: 1
    }]
},
options: {
    scales: {
        y: { beginAtZero: true }
    },
    onClick: function(event, elements) {
        if (elements.length > 0) {
            const clickedElementIndex = elements[0].index;
            const url = labels[clickedElementIndex];
            window.open(url, '_blank'); // Open the clicked URL in a new tab
        }
    }
}
});
}

```

```

</script>

</head>

<body>

<!--Navbar →

<div class="navbar">

  <a href="#home">Home</a>

  <a href="#shortener-section">URLs Shortener</a>

  <a href="#links-section">Dashboard</a>

  <a href="#visualization-section">Data Visualization</a> <!--New Link →

</div>

<!--Landing Page Section →

<div class="landing-page" id="home">

  <h1>Welcome to the URL Shortener <br>with QR Code and <br>Visualization
Features!</h1>

  <p>Shorten your URLs quickly and track the clicks for each of your links.</p>

  <a href="#shortener-section" class="cta-button">Get Started</a>

</div>

<!--Link Shortener Section →

<div class="container form-container" id="shortener-section">

```

```
<h1 class="heading">Link Shortener</h1>
```

```
<div class="form">
```

```
<form action="/shortUrls" method="post" onsubmit="showAlert()">
```

```
<input type="url" name="full" placeholder="Enter your link" required>
```

```
<button type="submit">Submit</button>
```

```
</form>
```

```
</div>
```

```
</div>
```

```
<!--Links Section -->
```

```
<div class="30ontainer links-container" id="links-section">
```

```
<h1 class="headings">Your Links</h1>
```

```
<div class="links">
```

```
<div class="link-box">
```

```
<div class="full">
```

```
<h2>Full Link</h2>
```

```
</div>
```

```
<div class="short">
```

```
<h2>Short Link & QR Code</h2>
```

```
</div>
```

```

    <div class="click">

    <h2>Clicks/Action</h2>

    </div>

</div>

</div>

<% shorturls.forEach((url, index) => { %>

    <div class="links">

        <div class="link-box">

            <div class="full">

                <a href="<%= url.full %>"><%= url.full %></a>

            </div>

            <div class="short">

                <a target="_blank" href="<%= url.short %>"><%= url.short %></a>

                <div id="qrcode-<%= index %>" style="display: inline-block;"></div>

                <!--Button to generate QR code -->

                <button id="qr-btn-<%= index %>" onclick="generateQRCode('<%= url.short %>',
<%= index %>)">Generate QR</button>

            </div>

            <div class="clicks">

                <span><%= url.clicks %></span>

            </div>

```

```

<div class="delete">

  <a href="/delete/<%= url._id %>">

    <span class="material-symbols-outlined">

      auto_delete

    </span>

  </a>

</div>

</div>

</div>

</div>

<%= }); %>

</div>

```

<!--Data Visualization Section -->

```

<div class="32ontainer visualization-container" id="visualization-section">

  <h1 class="headings">Click Data Visualization</h1>

  <canvas id="clickChart" width="700" height="200"></canvas>

</div>

```

<script>

```

// Fetching short URLs and click data dynamically

```

```

const shortUrls = [<%= shorturls.forEach(url => { %> "<%= url.short %>", <%= }); %>];

```



```
const clickCounts = [<% shorturls.forEach(url => { %> <%= url.clicks %>, <% }) %>];

// Data for the chart, using short URLs as labels
const clickData = {
  labels: shortUrls,
  datasets: [{
    label: 'Number of Clicks',
    data: clickCounts,
    backgroundColor: [
      'rgba(75, 192, 192, 0.2)',
      'rgba(153, 102, 255, 0.2)',
      'rgba(255, 159, 64, 0.2)',
      'rgba(255, 99, 132, 0.2)',
    ],
    borderColor: [
      'rgba(75, 192, 192, 1)',
      'rgba(153, 102, 255, 1)',
      'rgba(255, 159, 64, 1)',
      'rgba(255, 99, 132, 1)',
    ],
    borderWidth: 1
  }]
}
```

```

};

const config = {
  type: 'bar',
  data: clickData,
  options: {
    scales: {
      y: {
        beginAtZero: true
      }
    },
    onClick: function(event, elements) {
      if (elements.length > 0) {
        const clickedElementIndex = elements[0].index;
        const url = shortUrls[clickedElementIndex];
        window.open(url, '_blank'); // Open the clicked URL in a new tab
      }
    }
  }
};

const clickChart = new Chart(

```

```
        document.getElementById('clickChart'),  
        config  
    );  
</script>
```

```
</body>
```

```
</html>
```

Style.css:

```
/* Basic reset */
```

```
* {  
    margin: 0;  
    padding: 0;  
    box-sizing: border-box;  
}
```

```
body {  
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;  
    margin: 0;  
    padding: 0;  
    overflow-y: auto;  
    /* Allows scrolling */
```

```
}
```

```
/* Basic styles for landing page */
```

```
.landing-page {
```

```
  height: 100vh;
```

```
  display: flex;
```

```
  flex-direction: column;
```

```
  justify-content: center;
```

```
  align-items: center;
```

```
  text-align: center;
```

```
  /* Adjusted to allow some background visibility */
```

```
  padding: 20px;
```

```
}
```

```
.landing-page h1 {
```

```
  font-size: 48px;
```

```
  margin-bottom: 20px;
```

```
  color: darkolivegreen;
```

```
}
```

```
.landing-page p {
```

```

    font-size: 20px;

    margin-bottom: 30px;

    color: darkcyan;
}

.cta-button {

    padding: 10px 20px;

    font-size: 18px;

    background-color: #007BFF;

    color: white;

    text-decoration: none;

    border-radius: 5px;

    transition: background-color 0.3s;
}

.cta-button:hover {

    background-color: #0056b3;
}

/* Navbar styles */

/* Navbar styles */

.navbar {

```

```
display: flex;

justify-content: flex-start;

background-color: mediumturquoise;

padding: 5px 10px;

position: fixed;

top: 0;

width: 100%;

z-index: 1000;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
```

```
.navbar a {

color: white;

text-decoration: none;

background-color: darkcyan;

font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;

padding: 10px 15px;

font-size: 18px;

text-align: center;

border: 2px solid whitesmoke;

/* Gray border around each item */

border-radius: 5px;
```

```
margin-right: 3px;

/* Added space between navbar contents */

transition: background-color 0.3s, border-color 0.3s;
}

.navbar a:hover {

background-color: wheat;

color: black;

border-color: whitesmoke;
}

/* Smooth scroll behavior */

html {

scroll-behavior: smooth;
}

/* Section styling */

.container {

padding: 100px 50px 50px;

height: 100vh;

width: 100%;

margin: auto;
```

```
        text-align: center;
    }

.container .heading {
    font-size: 62px;
    color: darkcyan;
}

.containererr {
    padding: 100px 50px 50px;
    min-height: 100vh;
    /* Changed height to min-height for flexibility */
    width: 100%;
    margin: auto;
    text-align: center;
}

/* Headings styling */
.containererr .headings {
    font-size: 62px;
    color: darkcyan;
}
```



```
/* Media Queries for responsiveness */

@media screen and (max-width: 1024px) {

    .containerr {

        padding: 80px 30px 30px;

        /* Reduce padding for medium screens */

    }

    .containerr .headings {

        font-size: 48px;

        /* Adjust font size for medium screens */

    }

}

@media screen and (max-width: 768px) {

    .containerr {

        padding: 60px 20px 20px;

        /* Further reduce padding for tablets */

    }

    .containerr .headings {

        font-size: 36px;
```

```

        /* Adjust font size for tablets */
    }
}

@media screen and (max-width: 480px) {
    .containerr {
        padding: 40px 15px 15px;

        /* Reduce padding for mobile devices */
    }

    .containerr .headings {
        font-size: 28px;

        /* Adjust font size for mobile devices */
    }
}

div.form {
    padding: 30px 0;
}

div.form form input {
    padding: 5px 30px;
}

```

```
width: 80%;  
font-size: 20px;  
box-sizing: border-box;  
}  
  
div.form form button {  
    border: none;  
    outline: none;  
    background-color: #007BFF;  
    padding: 10px 24px;  
    color: #fff;  
    font-size: 18px;  
    cursor: pointer;  
    transition: background-color 0.3s ease;  
}  
  
div.form form button:hover {  
    background-color: #522084;  
}  
  
/* link-box */  
.link-box {
```

```
display: flex;
flex-wrap: wrap;
justify-content: space-between;
background: bisque;
padding: 10px;
margin: 2px;
}
```

```
.link-box .full,
.link-box .short {
width: 40%;
word-wrap: break-word;
}
```

```
.link-box div a {
color: #264653;
text-decoration: none;
}
```

```
.link-box div a:hover {
text-decoration: underline;
}
```

```
.clicks {  
  height: 20px;  
  width: 20px;  
  background: #119888;  
  color: #fff;  
  border-radius: 50%;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  margin: 0 20px;  
}
```

```
.delete span {  
  color: #d11a2a;  
  font-size: 24px;  
  cursor: pointer;  
  transition: color 0.3s ease;  
}
```

```
.delete span:hover {  
  color: #a10f20;  
}
```

```
}
```

```
/* Scrollbar styling */
```

```
body::-webkit-scrollbar {
```

```
    width: 12px;
```

```
}
```

```
body::-webkit-scrollbar-thumb {
```

```
    background-color: #888;
```

```
    border-radius: 6px;
```

```
}
```

```
body::-webkit-scrollbar-thumb:hover {
```

```
    background-color: #555;
```

```
}
```

```
/* Responsive styling */
```

```
@media screen and (max-width: 768px) {
```

```
    .navbar a {
```

```
        font-size: 16px;
```

```
        padding: 10px;
```

```
}
```

```
.landing-page h1 {  
    font-size: 36px;  
}  
  
.landing-page p {  
    font-size: 16px;  
}  
  
.container .heading {  
    font-size: 48px;  
}  
  
.link-box .full,  
.link-box .short {  
    width: 100%;  
}  
  
.link-box {  
    flex-direction: column;  
    align-items: flex-start;  
}  
}
```

```
@media screen and (max-width: 480px) {
```

```
  .navbar a {
```

```
    font-size: 14px;
```

```
    padding: 8px;
```

```
  }
```

```
  .landing-page h1 {
```

```
    font-size: 28px;
```

```
  }
```

```
  .landing-page p {
```

```
    font-size: 14px;
```

```
  }
```

```
  .container .heading {
```

```
    font-size: 36px;
```

```
  }
```

```
  .form input {
```

```
    width: 100%;
```

```
    padding: 10px;
```

```
  }
```



```

.link-box {
    padding: 10px;
}

.link-box .full,
.link-box .short {
    width: 100%;
}
}

/* QR code styling */
.qr-code {
    width: 100px;

    /* Default width for QR code */

    height: 50px;

    /* Default height for QR code */

    margin: 0 10px;

    /* Add margin for spacing */
}

/* Responsive QR code styling for smaller screens */
@media screen and (max-width: 768px) {

```

```
.qr-code {  
  width: 80px;  
  /* Reduce size for tablets */  
  height: 80px;  
}  
}  
  
@media screen and (max-width: 480px) {  
  .qr-code {  
    width: 60px;  
    /* Further reduce size for mobile phones */  
    height: 60px;  
  }  
}
```

5.1.2 Backend:

Server.js:

```
const express = require('express')  
const mongoose = require('mongoose')  
  
const app = express()
```

```
const PORT = process.env.PORT || 3000

mongoose.connect('mongodb://127.0.0.1:27017/urlshorts')

const db = mongoose.connection;

db.on('error', () => {
  console.log('error');
})

db.once('open', () => {
  console.log("connected");
})

app.set('view engine', 'ejs')
app.use(express.static('public'))
app.use(express.json())
app.use(express.urlencoded({ extended: true }))

// link router

const urlRouter = require('./routes/urlRoute')
app.use('/', urlRouter)
```

```
App.listen(PORT, () => {  
  console.log("server is running ");  
})
```

5.2 Screen Shots/ Output:

5.2.1 Home page:

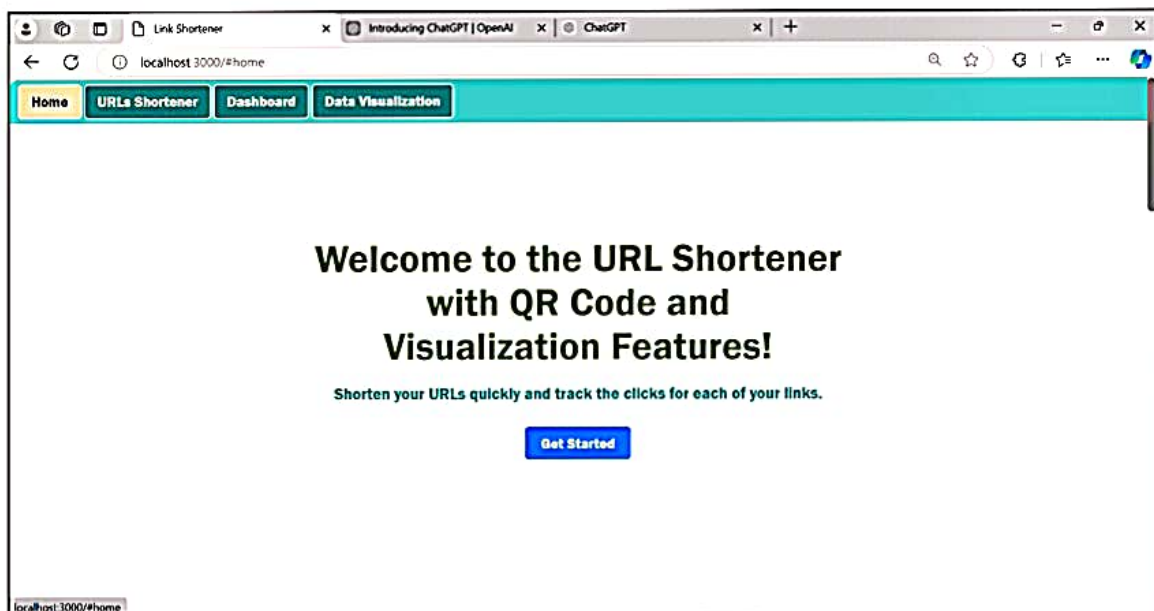
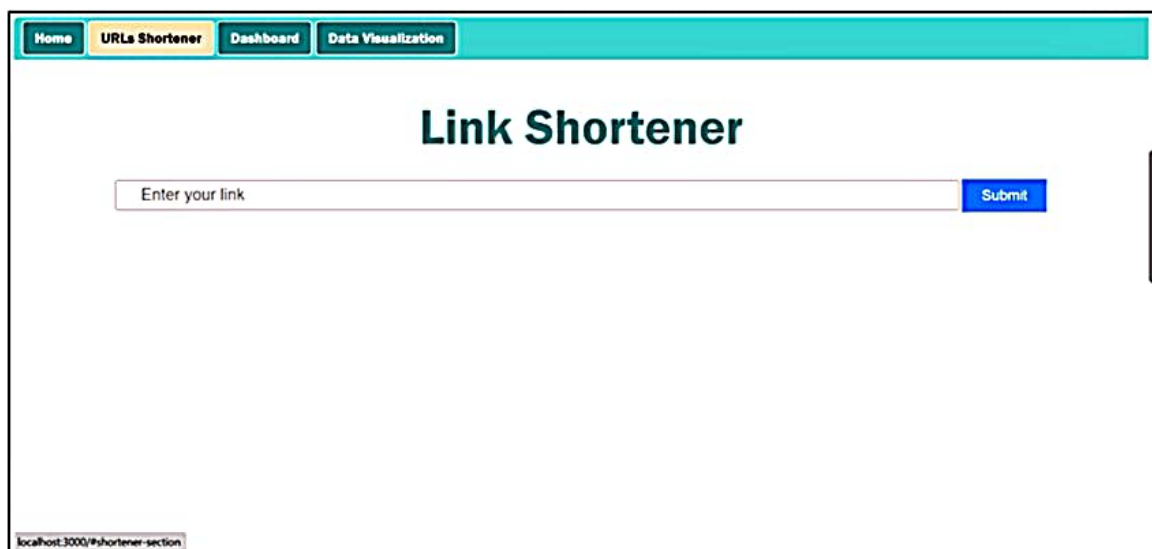


Figure 5.2.1

5.2.2 URL Shortener Interface:



The screenshot displays a web application interface for a URL shortener. At the top, there is a navigation bar with four tabs: "Home", "URLs Shortener" (which is highlighted in orange), "Dashboard", and "Data Visualization". Below the navigation bar, the main heading "Link Shortener" is centered. Underneath the heading is a form consisting of a text input field with the placeholder text "Enter your link" and a blue "Submit" button to its right. At the bottom left of the page, a small status bar shows the URL "localhost:3000/#shortener-section".

Figure 5.2.2

5.2.3 link Submitting Interface:



This screenshot shows the same URL Shortener interface as Figure 5.2.2, but with a successful submission. A modal dialog box is displayed in the center of the screen. The dialog has a title "localhost:3000 says" and a message "Your link has been submitted successfully!". It includes an "OK" button. In the background, the "Link Shortener" heading is visible, and the text input field now contains the URL "https://www.google.com". The "Submit" button remains to the right of the input field. The navigation bar and status bar are also visible.

Figure 5.2.3

5.2.4 Dashboard:

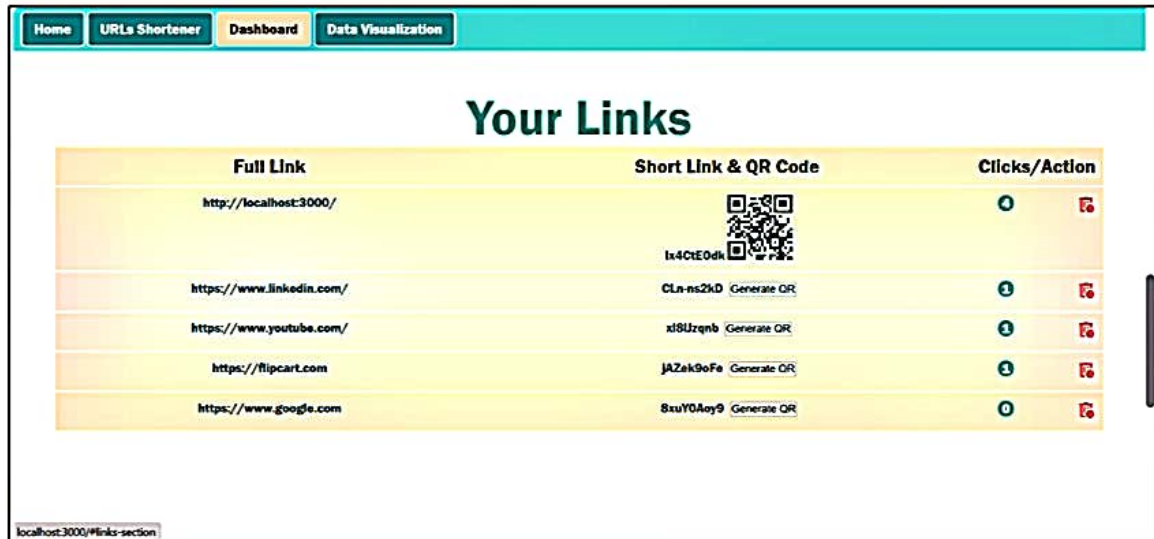


Figure 5.2.4

5.2.5 Click Analytics:

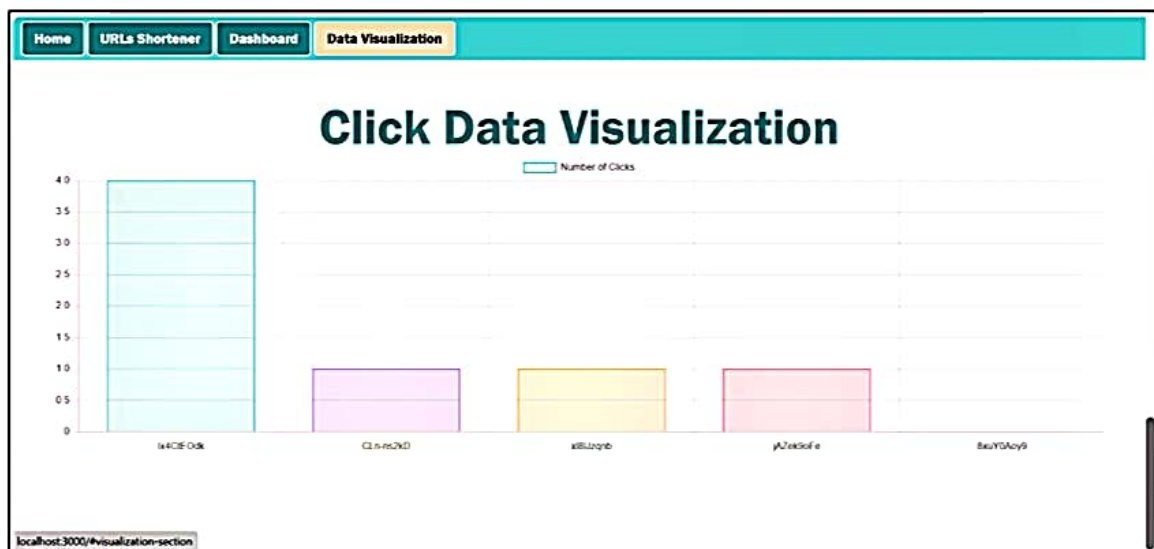


Figure 5.2.5

5.2.6 Data Base:

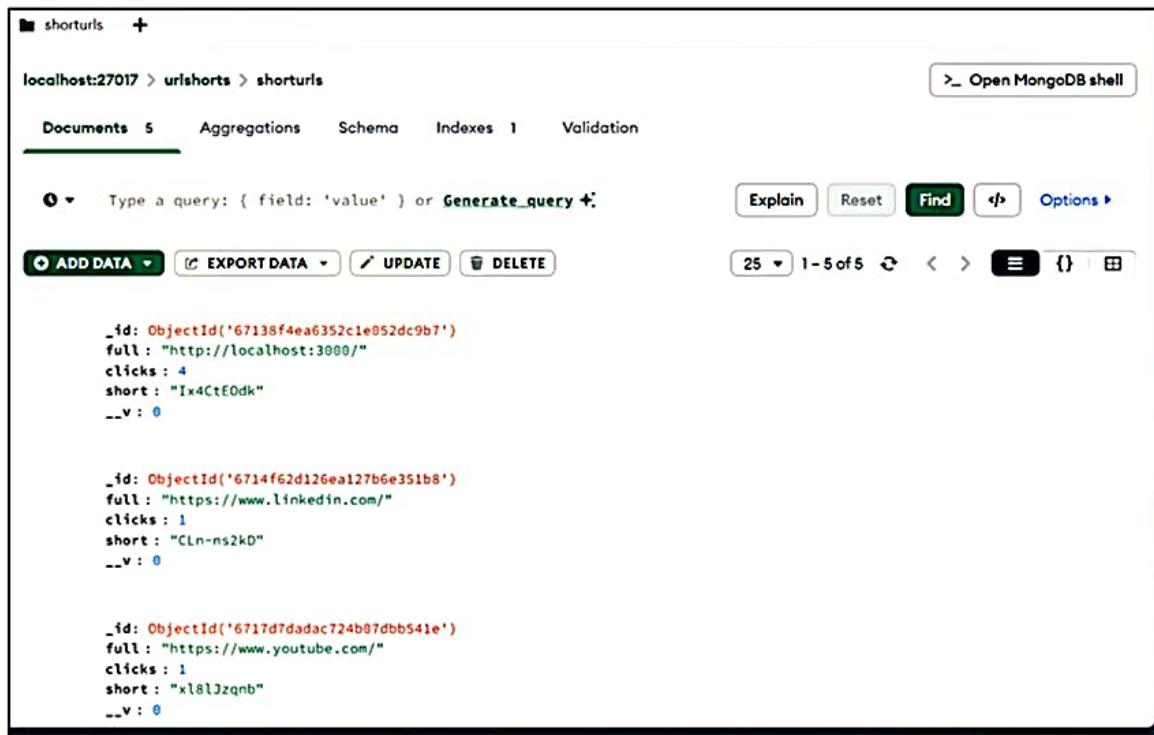


Figure 5.2.6

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion:

The **URL Shortener with QR Code and Visualization Features** project successfully meets its primary objectives, offering users a seamless and intuitive platform to manage long URLs, generate QR codes, and track link performance through real-time data visualization. The project addresses several practical challenges faced by users and organizations that frequently need to share long URLs, track engagement, and manage multiple links.

Key Achievements:

- **URL Shortening:** The system effectively shortens long URLs into compact, easy-to-share links. These links are stored in the database and can be easily retrieved or managed by users.
- **QR Code Generation:** For each shortened URL, the system automatically generates a corresponding QR code, which can be downloaded for easy sharing across physical and digital media.
- **Tracking and Visualization:** The system logs each visit to the shortened URLs, providing users with detailed analytics, including the number of visits and trends over time. The data is displayed using visually intuitive graphs and charts, enabling users to analyze the performance of their links.
- **User-Friendly Interface:** The use of **React.js** ensures that the system provides a smooth and responsive user experience. The interface is designed to be simple yet functional, making it easy for users to create, manage, and track their URLs.
- **Scalable and Efficient Backend:** The project utilizes **Node.js** and **Express.js** for the backend, ensuring that the system can handle a growing number of URLs and visits without performance degradation. The use of **MongoDB** allows for efficient data storage

and retrieval, making the system scalable for both personal and enterprise use.

Impact: This system solves the common problem of long URLs being difficult to share and track. It has direct applications in social media marketing, content management, and business analytics, among other fields. By integrating data visualization, users can gain valuable insights into how their URLs are being accessed and how often they are being engaged with, allowing for data-driven decision-making.

Overall, the project demonstrates the integration of modern web technologies to build a practical and scalable solution that simplifies the process of URL management. It is user-focused, efficient, and equipped with essential features for managing and tracking web links.

6.2 Future Enhancements:

While the current system is highly functional, there are several areas where future enhancements could be made to improve the overall system, add additional features, or expand its usability. These enhancements can make the system more flexible, secure, and feature-rich, catering to a wider range of use cases.

1. User Authentication and Authorization

- **Current Limitation:** At present, the system operates without user accounts, meaning any user can create, shorten, and manage URLs without authentication.
- **Proposed Enhancement:** Implement a user authentication system, where users can sign up, log in, and manage their URLs privately. This would allow users to have a personalized dashboard, storing and tracking their own set of URLs without interference from others.
- **Authorization Mechanism:** Roles like admin and regular user can be implemented to restrict certain functionalities (e.g., only admins can delete all URLs).

2. Enhanced Security Features

- **Current Limitation:** Although input validation is performed, there are opportunities to improve the security mechanisms of the system.
- **Proposed Enhancement:**
 - Implement measures to prevent **malicious URL submissions**, such as blacklisting certain domains or scanning URLs for malicious content before shortening them.
 - Add **CAPTCHA** to prevent bots from creating an excessive number of shortened URLs.
 - Use **rate limiting** to prevent abuse and spam, ensuring that a user cannot create a large number of shortened URLs in a short period.

3. Custom URL Aliases

- **Current Limitation:** The system automatically generates random shortened URLs, which may not always be memorable.
- **Proposed Enhancement:** Provide users with the option to create custom aliases for their shortened URLs. For example, a user could shorten <https://www.example.com/very-long-url> to <https://short.ly/customalias>, where "customalias" is chosen by the user. This feature would make URLs more user-friendly and easier to remember.
- This can also extend to **vanity URLs** for branding purposes, especially useful for businesses and marketing campaigns.

4. Advanced Analytics

- **Current Limitation:** The system tracks the number of visits and basic metadata (e.g., timestamps), but there is room for more granular data.
- **Proposed Enhancement:** Introduce more advanced analytics, such as:
 - **Geolocation data:** Track where the visits are coming from geographically.
 - **Device and browser data:** Analyze the devices and browsers being used to

access the shortened URLs.

- **Click-through rate (CTR):** Measure the percentage of users who engage with the QR code or shortened URL.
- These enhancements would provide users with a deeper understanding of the performance of their URLs.

5. Link Expiration and Scheduling

- **Current Limitation:** The shortened URLs created by the system are permanent and cannot be set to expire.
- **Proposed Enhancement:** Allow users to set expiration dates for their shortened URLs, after which the link would no longer be accessible. Users could also schedule links to become active at a future date and time. This feature would be particularly useful for time-sensitive content, such as promotions or limited-time offers.

6. Integration with Third-Party Services

- **Current Limitation:** The system operates as a standalone platform.
- **Proposed Enhancement:** Provide integration options with third-party services like **Google Analytics, Facebook Pixel, or HubSpot**. This would allow businesses to further track their URLs' performance across various platforms and measure engagement in more detail.

7. Mobile Application

- **Current Limitation:** The system is designed as a web-based application.
- **Proposed Enhancement:** Develop a **mobile application** for both iOS and Android platforms. A mobile app would provide users with the convenience of shortening URLs, generating QR codes, and tracking visits on the go. Mobile apps could also utilize device features like the camera to scan QR codes directly, enhancing the usability of the platform.