URL Shortener – Proof of Concept (PoC)

Name: Swadhin Das
Intern ID: 396
Organization: Digisuraksha Parhari Foundation
Project: URL Shortener using Node.js + SQLite + nanoid

Objective

To design a simple web-based **URL Shortener** that allows users to input a long URL and receive a unique short version for easy sharing and redirection.

Problem Statement (Simplified)

Long URLs are often hard to share, remember, or type — especially on social media or messaging platforms where space is limited. People want a way to make these links shorter, easier to manage, and more user-friendly.

The goal of this project is to build a simple tool that lets users:

- Enter a long URL
- Get a short version of that URL
- Click the short link and be redirected to the original one
- Record when the short link is used (for basic tracking)

This tool should be easy to use, run locally, and store data safely using a small database.

Tools Used	**Use Cases**
Node.js	Backend runtime
Express.js	Web framework
SQLite3	Lightweight local database
nanoid	Template engine for rendering views
EJS	Generates unique 6–8 character short slugs
HTML	Frontend

Project Structure:

```
url-shortner/
        public/
            index.html
        views/
            result.ejs
        server.js
        package.json
        urls.db
```

Use Case

A user wants to share a long URL on social media, but the link is too long and looks messy. The user opens the URL Shortener tool, pastes the long URL into a form, and clicks "Shorten." The tool gives a short, clean link that redirects to the original URL when clicked. The tool also tracks who clicked the short link and when.

Setup and Installation

Frontend Form

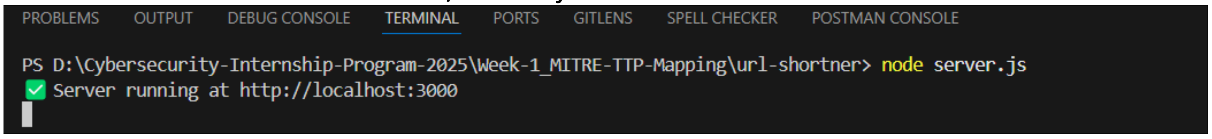
```
<form action="/shorten" method="POST">
  <input type="text" name="longUrl" placeholder="Enter long URL" required />
  <button type="submit">Shorten</button>
</form>
```

A simple HTML form that takes a long URL from the user and sends it to /shorten on the server.

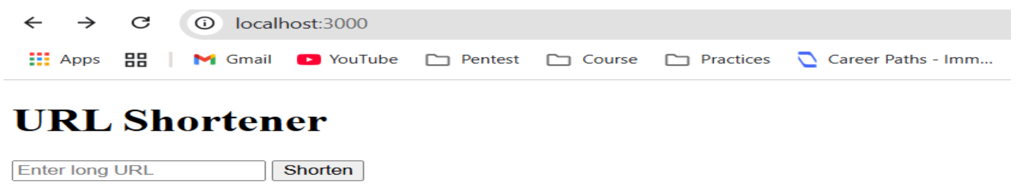
Backend Data

```
npm init -y
npm install express body-parser sqlite3 ejs nanoid
```

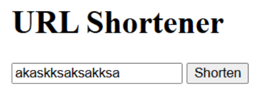
- Initializes a Node.js project.
- Installs required libraries:
  - express: web server
  - body-parser: handles form input
  - sqlite3: database to store URLs
  - ejs: displays output in a template
  - nanoid: generates secure 7-character short codes
- Create server.js
- Create Frontend HTML – public/index.html
- Create Result View – views/result.ejs



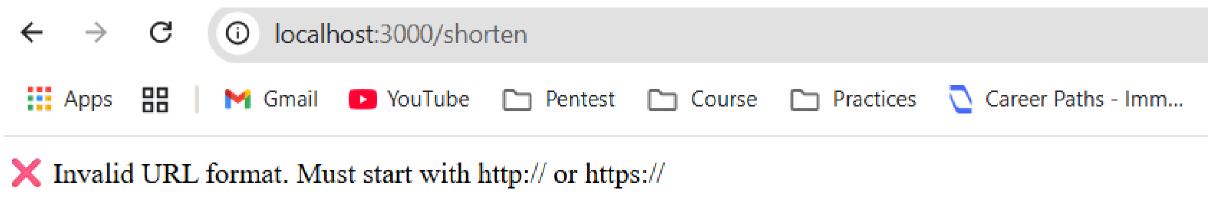
```
PS D:\Cybersecurity-Internship-Program-2025\Week-1_MITRE-TTP-Mapping\url-shortner> node server.js
✅ Server running at http://localhost:3000
```



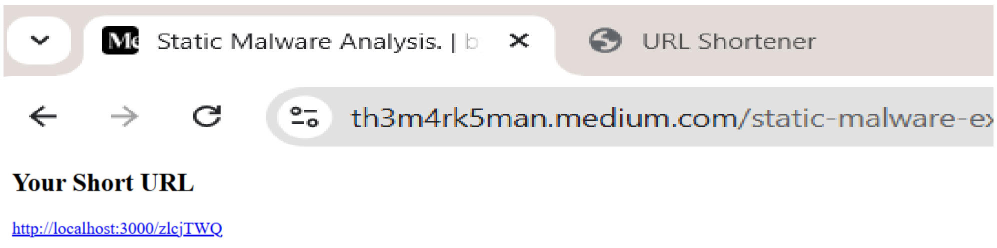
User have to give their long url input



If the given input url is invalid then it will show error



❌ Invalid URL format. Must start with http:// or https://


Static Malware Analysis. | b
URL Shortener
th3m4rk5man.medium.com/static-malware-e

Your Short URL

http://localhost:3000/zlcjTWQ

Database Setup

```
const sqlite3 = require('sqlite3').verbose();
const db = new sqlite3.Database('./urls.db');

db.serialize(() => {
  db.run('CREATE TABLE IF NOT EXISTS urls (id TEXT PRIMARY KEY, longUrl TEXT)');
  db.run('CREATE TABLE IF NOT EXISTS clicks (id INTEGER PRIMARY KEY AUTOINCREMENT,
shortId TEXT, timestamp TEXT, ip TEXT)');
});
```

Code: urls.db created via

- SQLite database file (urls.db) is created.
- urls table stores short code and corresponding long URL.
- clicks table logs each visit with timestamp and IP address.

Short Code Logic with nanoid

I use the nanoid package to generate a unique 7-character string for each URL.

```
const { nanoid } = require('nanoid');
const id = nanoid(7);
```

This short code is:

- Secure
- URL-safe
- Randomly generated

Shorten Route Logic

```
const { nanoid } = require('nanoid');

app.post('/shorten', (req, res) => {
  const longUrl = req.body.longUrl;
  const id = nanoid(7); // generates 7-character random short code

  db.run('INSERT INTO urls (id, longUrl) VALUES (?, ?)', [id, longUrl], (err) => {
    if (err) return res.send('DB Error');
    res.render('result', { shortUrl: `http://localhost:${PORT}/${id}` });
  });
});
```

When user submits the form:

- A 7-character random ID is created.
- It's stored in the DB with the original URL.
- A short URL is generated and displayed using EJS.

Result View (EJS)

```
<h2>Your Short URL</h2>
<a href="<%= shortUrl %>"><%= shortUrl %></a>
```

- Displays the short URL generated by the server.
- User can click to test redirection.

Redirection Logic

```
 app.get('/:id', (req, res) => {
  const id = req.params.id;

  db.get('SELECT longUrl FROM urls WHERE id = ?', [id], (err, row) => {
    if (err || !row) return res.send('URL not found!');

    const timestamp = new Date().toISOString();
    const ip = req.ip;

    db.run('INSERT INTO clicks (shortId, timestamp, ip) VALUES (?, ?, ?)', [id, timestamp, ip]);
    res.redirect(row.longUrl);
  });
});
```

When a short URL is opened:

- It looks up the original URL from the DB using the short code (id).
- Redirects to the original long URL.
- Also logs the visit into clicks table (IP & time).

User Flow

Here's a simple flow of how a user interacts with the tool:

1. **Visit Tool**: User opens http://localhost:3000.
2. **Enter URL**: User enters a long URL in the input box.
3. **Click Submit**: User clicks the "Shorten" button.
4. **Receive Short URL**: Tool displays a short link (e.g., http://localhost:3000/xYz1234).
5. **Share Link**: User copies and shares the short URL.
6. **Someone Clicks It**: When someone opens the short URL:
   - They're redirected to the original long URL.
   - The tool logs the time and IP address of the click.

Test Cases

Valid long URL submitted	-	Short URL is generated
Invalid URL	-	Error message shown
Visiting short URL	-	Redirects to original URL
Visiting invalid short code	-	"URL not found" message

Benefits of the Tool

Simplifies Link Sharing	-	Makes long URLs short, neat, and easy to share.
Secure & Unique	-	Each link is safely generated using nanoid, avoiding duplicates.
Stores Data Locally	-	Uses SQLite to save data without needing a server or cloud.

Easy to Maintain	-	Built using simple technologies (Node.js, Express.js, SQLite).
Ready for Deployment	-	Can be deployed online or integrated into larger applications.

Who Can Use the Tool?
This URL Shortener tool can be used by:

General Users

Shorten long URLs to make them easier to share on social media, emails, or SMS.

Students & Developers

Learn how web routing, databases, and redirection systems work.

Content Creators & Marketers

Share compact, clean links in campaigns and track user clicks.

Organizations

Build internal link tracking or branding solutions with custom short links.

Cybersecurity Interns

Understand how redirect-based phishing or tracking links might be misused, helping in awareness and detection.

The project uses simple tools like Node.js, Express, EJS, SQLite, and can run locally on your computer. It is useful for learning how:

- Web forms work
- Data is saved in a database

Redirects are handled on the backend

Conclusion

The URL Shortener project successfully demonstrates the design and working of a full-stack web application , shows how a basic URL Shortener works. It takes a long URL from the user and creates a short link using a random 7-character code (with nanoid). The tool stores both URLs in a local SQLite database.
When someone clicks the short link:

- They are redirected to the original URL.
- The tool logs the click with the time and IP address.

It serves as an excellent learning project in web development and cybersecurity, covering key concepts like:

- Data storage
- Random ID generation
- Routing