📘 Report: URL Shortener Project

🚀 Project Title: URL Shortener (Flask + SQLite)

💡 Objective:

To design and implement a basic web-based URL shortener (similar to bit.ly or tinyurl) that can take a long URL and return a shortened version for easy sharing and redirection.

📌 Features Implemented:

- Web form to input a long URL
- Flask-based backend API
- Auto-generated short URL using Base62 encoding
- SQLite database to store long and short URL mappings
- Redirection functionality when the short URL is accessed

🧪 Tools & Technologies Used:

- **Language:** Python 3.x
- **Framework:** Flask
- **Database:** SQLite
- **Editor:** Visual Studio Code (VS Code)
- **Version Control:** Git & GitHub

🧠 How It Works:

1. The user inputs a long URL into a form on the homepage.
2. Flask receives this input and stores the URL in a SQLite database.
3. The database auto-generates an ID which is then encoded to a Base62 string to form a unique short code.

4. The short code is appended to the host URL and shown to the user.
5. When the user clicks the short URL, Flask retrieves the original URL from the database and redirects the browser.

⚙️ How to Run This Project

🛠️ Follow these steps in your terminal or VS Code:

🔹 **Step 1: Clone the Repository**

```bash
git clone https://github.com/yourusername/url-shortener.git
cd url-shortener
```

🔹 Step 2: Create and Activate a Virtual Environment

```bash
# For Windows
python -m venv venv
venv\Scripts\activate

# For Mac/Linux
python3 -m venv venv
source venv/bin/activate
```

🔹 Step 3: Install Flask

```bash
pip install flask
```

🔹 Step 4: Run the Application

```
bash

python app.py
```

🔹 Step 5: Open Your Browser

Visit:

```
cpp

http://127.0.0.1:5000
```

Try submitting a long URL and test the shortened version.

📁 **Folder Structure:**

```
bash

url_shortener/
├── app.py                  # Main backend logic
├── shortener.db            # Auto-created SQLite database
├── templates/
│   └── index.html          # HTML form
└── venv/                   # Python virtual environment
```

📉 **Disadvantages / Limitations**

Limitation	**Description**
❌ **No user authentication**	**Anyone can use it without control or tracking**
❌ **No analytics**	**No way to see how many times a short URL was clicked**
❌ **No custom short codes**	**Users cannot define their own slugs**

Limitation	Description
❌ Not production-ready	This is a local, basic implementation — not scalable
❌ No input validation	Does not yet check if the URL is valid or malicious

---

⚠️ Disclaimer

This project was developed during my internship for educational purposes only.
Some guidance and partial code structure were supported by KaliGPT — an AI assistant specializing in cybersecurity and development workflows.

All source code has been understood, reviewed, and tested by me personally. The solution was kept simple for learning and demonstration purposes within the organization.

✅ Future Improvements

- Add URL validation (to reject bad input)
- Add login/auth system for personal links
- Deploy using Docker or a cloud platform (Render/Heroku)
- Add analytics (click count per short link)
- Set expiry dates for short URLs