Intern Name: Aditya Yashwant Borade **Intern ID** : 131
Organization: Digisurakhsha Foundation

Weekly Task Document 2: URL Shortener

✅ Objective:

Design and implement a basic web-based **URL Shortener** that accepts a long URL and returns a **shortened link** (like bit.ly, tinyurl.com). The service should allow redirection using the shortened URL.

🔧 Rough Idea / Starting Points:

- Use **Flask (Python)** for the backend.
- Use **SQLite** or **in-memory dictionary** for mapping storage.
- Generate short codes using:
    - Base62 encoding of auto-incrementing IDs, or
    - Random string generator (recommended for simplicity).

🔄 System Architecture Overview

[User Input (Long URL)]

↓

[Frontend Form (HTML)]

↓

[Flask Backend API]

↓

[Shortcode Generation]

↓

[Store in SQLite or Dictionary]

↓

[Return Short URL to User]

↓

[On Access → Redirect to Original URL]

💻 Implementation (Flask + SQLite)

1. Install Dependencies

```
pip install flask flask_sqlalchemy
```

---

2. Flask Code (app.py)

```
from flask import Flask, request, redirect, jsonify
from flask_sqlalchemy import SQLAlchemy
import string, random

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///urls.db'
db = SQLAlchemy(app)

class URLMap(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    long_url = db.Column(db.String(500), nullable=False)
    short_code = db.Column(db.String(8), unique=True, nullable=False)

def generate_short_code(length=6):
    characters = string.ascii_letters + string.digits
    return ''.join(random.choices(characters, k=length))

@app.before_first_request
def create_tables():
    db.create_all()

@app.route('/shorten', methods=['POST'])
def shorten_url():
    data = request.get_json()
    long_url = data.get("long_url")
    short_code = generate_short_code()
    while URLMap.query.filter_by(short_code=short_code).first():
        short_code = generate_short_code()

    new_url = URLMap(long_url=long_url, short_code=short_code)
    db.session.add(new_url)
    db.session.commit()

    return jsonify({"short_url": request.host_url + short_code})

@app.route('/<short_code>')
def redirect_to_long_url(short_code):
    url_entry = URLMap.query.filter_by(short_code=short_code).first()
    if url_entry:
        return redirect(url_entry.long_url)
    return "URL not found", 404

if __name__ == '__main__':
    app.run(debug=True)
```

---

3. Frontend (HTML Form)

```
<!DOCTYPE html>
<html>
<head>
  <title>URL Shortener</title>
</head>
<body>
  <h2>Enter a Long URL to Shorten</h2>
  <form id="urlForm">
    <input type="text" id="longUrl" placeholder="Enter URL" required>
    <button type="submit">Shorten</button>
  </form>
  <p id="result"></p>

  <script>
    document.getElementById('urlForm').onsubmit = async function(e) {
      e.preventDefault();
      const longUrl = document.getElementById('longUrl').value;
      const response = await fetch('/shorten', {
        method: 'POST',
        headers: {'Content-Type': 'application/json'},
        body: JSON.stringify({long_url: longUrl})
      });
      const data = await response.json();
      document.getElementById('result').innerText = "Short URL: " + data.short_url;
    };
  </script>
</body>
</html>
```

🧪 **Example Output**

1. **Input:**

```
https://www.wikipedia.org
```

2. **Generated Short Code:**

```
abc123
```

3. **Short URL Response:**

```
http://localhost:5000/abc123
```

4. **On Accessing http://localhost:5000/abc123:**
   → Redirects to https://www.wikipedia.org

✅ **Key Features**

Feature	**Description**
Shorten Long URL	Accepts long URLs and returns shortened links
Redirection	Redirects short links to original URLs
Unique Code Generator	Uses alphanumeric random strings
Persistent Storage	Stores data using SQLite