Python Project 13: URL Shortener

1. Project Overview

Build a simple desktop **URL Shortener** that converts long URLs into short local links and stores them in an SQLite database.

Example:

Long URL:
https://www.example.com/articles/python/programming

Short URL:
http://localhost:5000/aB3xK9

This project uses SQLite for persistent storage. Python's `sqlite3` module supports parameterized SQL queries using placeholders, which is used here instead of building SQL queries with string formatting.

2. Features

- Enter a long URL
- Generate a random short code
- Store URLs in SQLite
- View saved URLs
- Copy short URL
- Delete saved URLs
- Prevent duplicate short codes
- Validate HTTP and HTTPS URLs
- Simple Tkinter interface
- No external packages

3. Technologies

- Python 3
- Tkinter
- SQLite
- sqlite3
- secrets
- urllib.parse
- Object Oriented Programming

urllib.parse provides URL parsing functions that can be used to inspect URL components such as the scheme and network location.

4. Folder Structure

```
url_shortener/
□□□ main.py
□□□ ui.py
□□□ database.py
□□□ shortener.py
□□□ README.md
```

5. How the Project Works

```
User enters long URL
        □
Validate URL
        □
Generate short code
        □
Save URL + code
        □
Display short URL
        □
Store data in SQLite
```

The short code is randomly generated using Python's secrets module.

6. Complete Backend Code

shortener.py

```
import secrets
import string
from urllib.parse import urlparse


class URLShortener:

    CHARACTERS = string.ascii_letters + string.digits

    def __init__(self, database):
        self.database = database

    def validate_url(self, url):
        parsed = urlparse(url)

        return (
            parsed.scheme in ("http", "https")
            and bool(parsed.netloc)
        )

    def generate_code(self, length=6):
        return "".join(
            secrets.choice(self.CHARACTERS)
            for _ in range(length)
        )

    def shorten(self, url):
        url = url.strip()

        if not self.validate_url(url):
```

```
            raise ValueError(
                "Please enter a valid HTTP or HTTPS URL."
            )

        existing = self.database.get_by_url(url)

        if existing:
            return existing[1]

        while True:
            code = self.generate_code()

            if not self.database.get_by_code(code):
                break

        self.database.add_url(url, code)

        return code

    def get_short_url(self, code):
        return f"http://localhost:5000/{code}"
```

7. Complete Database Code

database.py

```
import sqlite3


class URLDatabase:

    def __init__(self, database_name="urls.db"):
        self.database_name = database_name
        self.create_table()
```

```
    def connect(self):
        return sqlite3.connect(self.database_name)

    def create_table(self):
        connection = self.connect()
        cursor = connection.cursor()

        cursor.execute("""
            CREATE TABLE IF NOT EXISTS urls (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                original_url TEXT NOT NULL,
                short_code TEXT UNIQUE NOT NULL
            )
        """)

        connection.commit()
        connection.close()

    def add_url(self, original_url, short_code):
        connection = self.connect()
        cursor = connection.cursor()

        cursor.execute("""
            INSERT INTO urls
            (original_url, short_code)
            VALUES (?, ?)
        """, (original_url, short_code))

        connection.commit()
        connection.close()

    def get_by_url(self, original_url):
        connection = self.connect()
        cursor = connection.cursor()
```

```
        cursor.execute("""
            SELECT id, short_code
            FROM urls
            WHERE original_url = ?
        """, (original_url,))

        result = cursor.fetchone()

        connection.close()

        return result

    def get_by_code(self, short_code):
        connection = self.connect()
        cursor = connection.cursor()

        cursor.execute("""
            SELECT id, original_url, short_code
            FROM urls
            WHERE short_code = ?
        """, (short_code,))

        result = cursor.fetchone()

        connection.close()

        return result

    def get_all(self):
        connection = self.connect()
        cursor = connection.cursor()

        cursor.execute("""
```

```
            SELECT id, original_url, short_code
            FROM urls
            ORDER BY id DESC
        """)

        results = cursor.fetchall()

        connection.close()

        return results

    def delete_url(self, url_id):
        connection = self.connect()
        cursor = connection.cursor()

        cursor.execute("""
            DELETE FROM urls
            WHERE id = ?
        """, (url_id,))

        connection.commit()
        connection.close()
```

8. Complete Frontend Code

ui.py

```
import tkinter as tk
from tkinter import messagebox, ttk

from database import URLDatabase
from shortener import URLShortener


class URLShortenerUI:
```

```
    def __init__(self, root):

        self.root = root
        self.root.title("URL Shortener")
        self.root.geometry("900x600")
        self.root.configure(bg="#0f172a")

        self.database = URLDatabase()
        self.shortener = URLShortener(
            self.database
        )

        self.url_var = tk.StringVar()

        self.create_ui()
        self.load_urls()

    def create_ui(self):

        title = tk.Label(
            self.root,
            text="URL Shortener",
            font=("Arial", 26, "bold"),
            bg="#0f172a",
            fg="white"
        )
        title.pack(pady=(30, 5))

        subtitle = tk.Label(
            self.root,
            text="Turn long URLs into short codes",
            font=("Arial", 11),
            bg="#0f172a",
```

```
            fg="#94a3b8"
        )
        subtitle.pack()

        input_frame = tk.Frame(
            self.root,
            bg="#0f172a"
        )
        input_frame.pack(
            fill="x",
            padx=50,
            pady=25
        )

        url_entry = tk.Entry(
            input_frame,
            textvariable=self.url_var,
            font=("Arial", 12),
            bg="#1e293b",
            fg="white",
            insertbackground="white",
            relief="flat"
        )

        url_entry.pack(
            side="left",
            fill="x",
            expand=True,
            ipady=10,
            padx=(0, 10)
        )

        url_entry.bind(
            "<Return>",
```

```
            lambda event: self.shorten_url()
        )

        shorten_button = tk.Button(
            input_frame,
            text="Shorten URL",
            command=self.shorten_url,
            font=("Arial", 11, "bold"),
            bg="#2563eb",
            fg="white",
            relief="flat",
            padx=20,
            pady=9
        )

        shorten_button.pack(side="right")

        self.result_label = tk.Label(
            self.root,
            text="Enter a URL to begin.",
            font=("Arial", 12),
            bg="#0f172a",
            fg="#94a3b8"
        )

        self.result_label.pack(pady=(0, 20))

        table_frame = tk.Frame(
            self.root,
            bg="#1e293b"
        )

        table_frame.pack(
            fill="both",
```

```
            expand=True,
            padx=50
        )

        columns = (
            "id",
            "original",
            "short"
        )

        self.tree = ttk.Treeview(
            table_frame,
            columns=columns,
            show="headings"
        )

        self.tree.heading(
            "id",
            text="ID"
        )

        self.tree.heading(
            "original",
            text="Original URL"
        )

        self.tree.heading(
            "short",
            text="Short Code"
        )

        self.tree.column(
            "id",
            width=50
```

```
        )

        self.tree.column(
            "original",
            width=600
        )

        self.tree.column(
            "short",
            width=150
        )

        scrollbar = ttk.Scrollbar(
            table_frame,
            orient="vertical",
            command=self.tree.yview
        )

        self.tree.configure(
            yscrollcommand=scrollbar.set
        )

        self.tree.pack(
            side="left",
            fill="both",
            expand=True
        )

        scrollbar.pack(
            side="right",
            fill="y"
        )

        button_frame = tk.Frame(
```

```
            self.root,
            bg="#0f172a"
        )

        button_frame.pack(pady=20)

        copy_button = tk.Button(
            button_frame,
            text="Copy Short URL",
            command=self.copy_short_url,
            bg="#334155",
            fg="white",
            relief="flat",
            padx=15,
            pady=8
        )

        copy_button.pack(
            side="left",
            padx=5
        )

        delete_button = tk.Button(
            button_frame,
            text="Delete",
            command=self.delete_url,
            bg="#334155",
            fg="white",
            relief="flat",
            padx=15,
            pady=8
        )

        delete_button.pack(
```

```
            side="left",
            padx=5
        )

    def shorten_url(self):

        url = self.url_var.get().strip()

        if not url:
            messagebox.showwarning(
                "Missing URL",
                "Please enter a URL."
            )
            return

        try:
            code = self.shortener.shorten(url)

            short_url = self.shortener.get_short_url(
                code
            )

            self.result_label.config(
                text=f"Short URL: {short_url}",
                fg="#60a5fa"
            )

            self.url_var.set("")

            self.load_urls()

        except ValueError as error:

            messagebox.showerror(
```

```
            "Invalid URL",
            str(error)
        )

    except Exception as error:

        messagebox.showerror(
            "Error",
            str(error)
        )

def load_urls(self):

    for item in self.tree.get_children():
        self.tree.delete(item)

    urls = self.database.get_all()

    for url_id, original, code in urls:

        self.tree.insert(
            "",
            "end",
            values=(
                url_id,
                original,
                code
            )
        )

def copy_short_url(self):

    selected = self.tree.selection()
```

```
        if not selected:
            messagebox.showwarning(
                "No Selection",
                "Please select a URL first."
            )
            return

        values = self.tree.item(
            selected[0],
            "values"
        )

        code = values[2]

        short_url = self.shortener.get_short_url(
            code
        )

        self.root.clipboard_clear()
        self.root.clipboard_append(short_url)

        self.result_label.config(
            text="Short URL copied to clipboard.",
            fg="#60a5fa"
        )

    def delete_url(self):

        selected = self.tree.selection()

        if not selected:
            messagebox.showwarning(
                "No Selection",
                "Please select a URL first."
```

```
            )
            return

        values = self.tree.item(
            selected[0],
            "values"
        )

        url_id = values[0]

        confirm = messagebox.askyesno(
            "Delete URL",
            "Delete the selected URL?"
        )

        if confirm:
            self.database.delete_url(url_id)
            self.load_urls()
            self.result_label.config(
                text="URL deleted."
            )
```

9. Main Code

main.py

```
import tkinter as tk

from ui import URLShortenerUI


def main():

    root = tk.Tk()
```

```
    URLShortenerUI(root)

    root.mainloop()


if __name__ == "__main__":
    main()
```

10. README

README.md

```
# URL Shortener

A simple Python desktop application that converts
long URLs into short codes.

## Features

- Generate short codes
- Store URLs in SQLite
- View saved URLs
- Copy short URLs
- Delete URLs
- URL validation

## Requirements

Python 3

No external packages are required.

## Run

python main.py
```

11. How Frontend + Backend Connect

ui.py

URLShortener

Validate URL

Generate unique code

URLDatabase

SQLite

Return short code

Display in UI

The database uses parameterized SQL placeholders such as ? when inserting and querying values rather than concatenating user input into SQL statements.

12. How to Run

Open the project directory:

```
cd url_shortener
```

Run:

```
python main.py
```

Enter a URL such as:

`https://www.python.org/`

Click **Shorten URL**.

The application generates a result similar to:

`http://localhost:5000/aB3xK9`

13. Basic Testing

Test these URLs:

`https://www.python.org/`
`https://github.com/`
`https://www.google.com/`

Also test:

- Empty URL
- Invalid URL
- URL without `http://` or `https://`
- Same URL multiple times
- Multiple different URLs
- Selecting and copying a short URL
- Deleting a saved URL

14. Important Project Limitation

This version is a **desktop URL shortening system**, so the generated `localhost` URL is a representation of the short code rather than a publicly accessible redirect service.

To turn it into a real URL shortener, the next version would need a web backend such as Flask or FastAPI with a route like:

`/<short_code>`

That route would look up the code in the database and redirect the visitor to the original URL.

15. Small Improvements

- Build a Flask/FastAPI web version
- Add real redirects
- Add click statistics
- Add custom short codes
- Add expiration dates
- Add QR code generation
- Add password protected links
- Add URL analytics
- Deploy the API online

Visit haas.dev for more resources and guides.