JOB AGGREGATOR LITE

PROJECT 6

Jack Kammerer

CS.4332.252 Introduction to Database Systems

Texas State University

March 2025

Contents

1 Introduction

This document outlines the design and implementation details for **Job Aggregator Lite**, a web scraper application built for Project 6. The project includes both a functional web scraper and a preliminary database design to store scraped data.

2 Business Forms & Requirement Analysis

2.1 Business Forms and UI

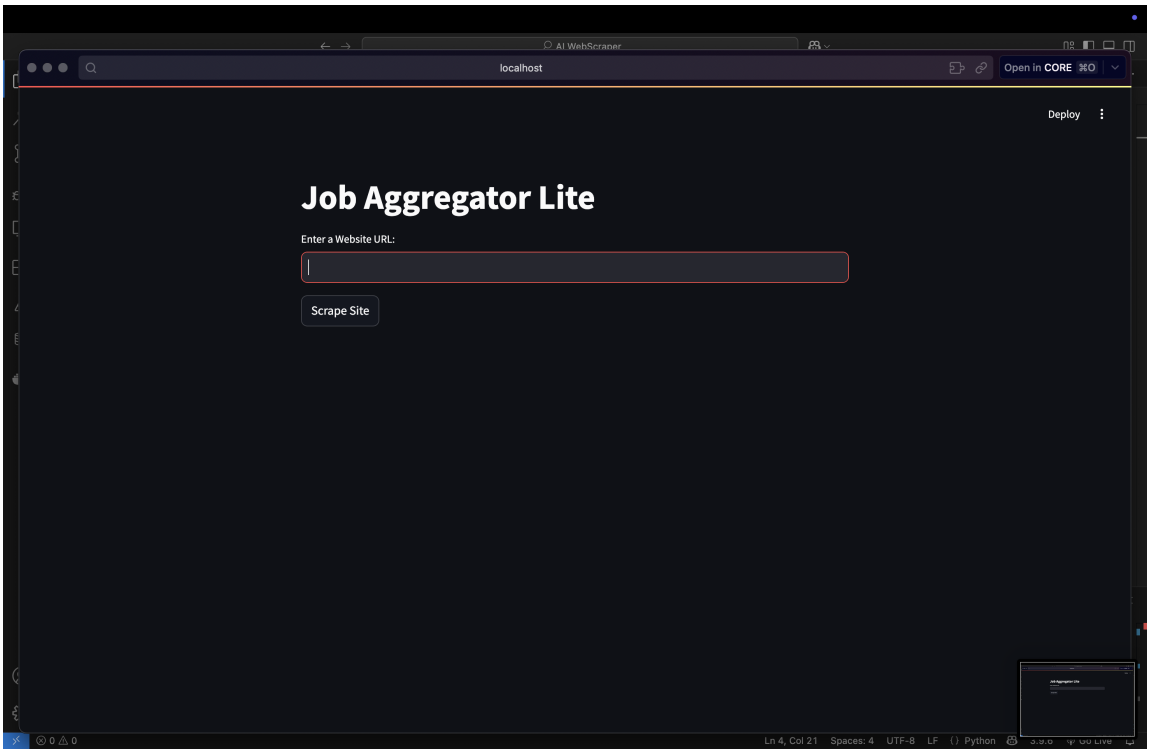

Figure 1: Webscraper UI

- **Input Form:** A simple UI where users enter a URL to be scraped.
- **Results Display:** Once the scraper runs, it shows the scraped content, including details like the URL, timestamp, and content type.
- **History/Report Page:** A page listing previous scrapes with their details.

2.2 Summary of Requirements

- The user enters a URL like Indeed or LinkedIn

- The system scrapes data (e.g., text, links) from the provided URL.
- The scraped data is stored in a relational database.
- A history page lets users view past scrapes.

3 User View Schema

3.1 User Journey

1. **Home Page:** User inputs the URL.
2. **Scraping Process:** The system scrapes the URL.
3. **Results Page:** Displays the scraped data including content type and timestamp.
4. **History Page:** Shows past scrapes for review.

3.2 Schema Diagram

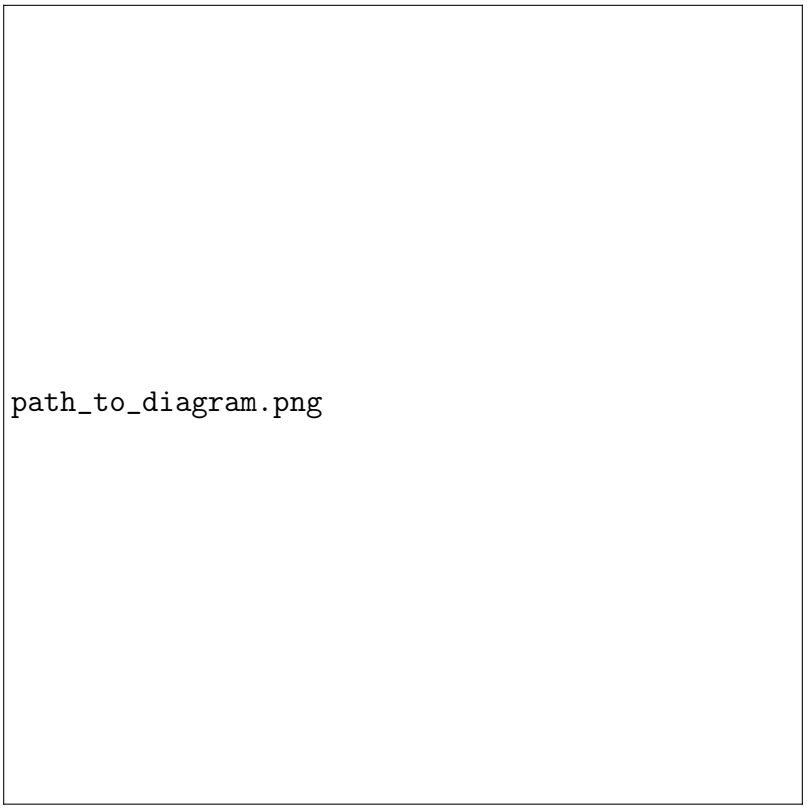

Figure 2: User View Schema Diagram

4 Preliminary Database Formation

4.1 Database Schema

The database consists of two main tables:

Websites Table

- **Columns:** `website_id (PK)`, `url`, `date_added`
- **Functional Dependency:** `website_id` $\rightarrow$ `url, date_added`

Scraped Data Table

- **Columns:** `data_id (PK)`, `website_id (FK)`, `content_type`, `content`, `date_scraped`
- **Functional Dependency:** `data_id` $\rightarrow$ `website_id, content_type, content, date_scraped`

4.2 Normalization

Both tables are designed in Third Normal Form (3NF):

- All non-key attributes are fully functionally dependent on the primary key.
- There are no transitive dependencies.

4.3 Sample Data

Websites Table:

```
website_id | url                          | date_added
-----------------------------------------------------
1          | https://www.example.com      | 2025-03-27 10:00 AM
2          | https://www.testsite.org     | 2025-03-28 02:15 PM
```

Scraped Data Table:

```
data_id | website_id | content_type | content
-------------------------------------------------------------------------------
1001    | 1          | text         | "Welcome to Example.com, we offer..."
1002    | 1          | link         | "https://www.example.com/about"
1003    | 2          | text         | "Test Site: This is a sample paragraph..."
```

4.4 Code Integration

Below is an excerpt of the database integration code from `db.py`:

```
import sqlite3
from datetime import datetime

def init_db():
    conn = sqlite3.connect("scraper.db")
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS websites (
                    website_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    url TEXT UNIQUE,
                    date_added DATETIME)''')
    c.execute('''CREATE TABLE IF NOT EXISTS scraped_data (
                    data_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    website_id INTEGER,
                    content_type TEXT,
                    content TEXT,
                    date_scraped DATETIME,
                    FOREIGN KEY (website_id) REFERENCES websites (
    website_id))''')
    conn.commit()
    conn.close()

def insert_scrape(url, scraped_items):
    conn = sqlite3.connect("scraper.db")
    c = conn.cursor()
    now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    # Insert into websites table if new URL
    c.execute("INSERT OR IGNORE INTO websites (url, date_added) VALUES (?,
    ?)", (url, now))
    conn.commit()

    # Get website_id
    c.execute("SELECT website_id FROM websites WHERE url = ?", (url,))
    website_id = c.fetchone()[0]

    # Insert scraped items (assumes scraped_items is a list of tuples: (
    content_type, content))
    for item in scraped_items:
        c.execute("INSERT INTO scraped_data (website_id, content_type,
    content, date_scraped) VALUES (?, ?, ?, ?)",
```

```
                    (website_id, item[0], item[1], now))
    conn.commit()
    conn.close()
```

Listing 1: Database Code in db.py

5 Conclusion

This document has detailed the design of Job Aggregator Lite, covering the UI, user flow, and preliminary database design. Future work involves fully integrating the UI with the database and performing thorough testing to ensure smooth functionality.