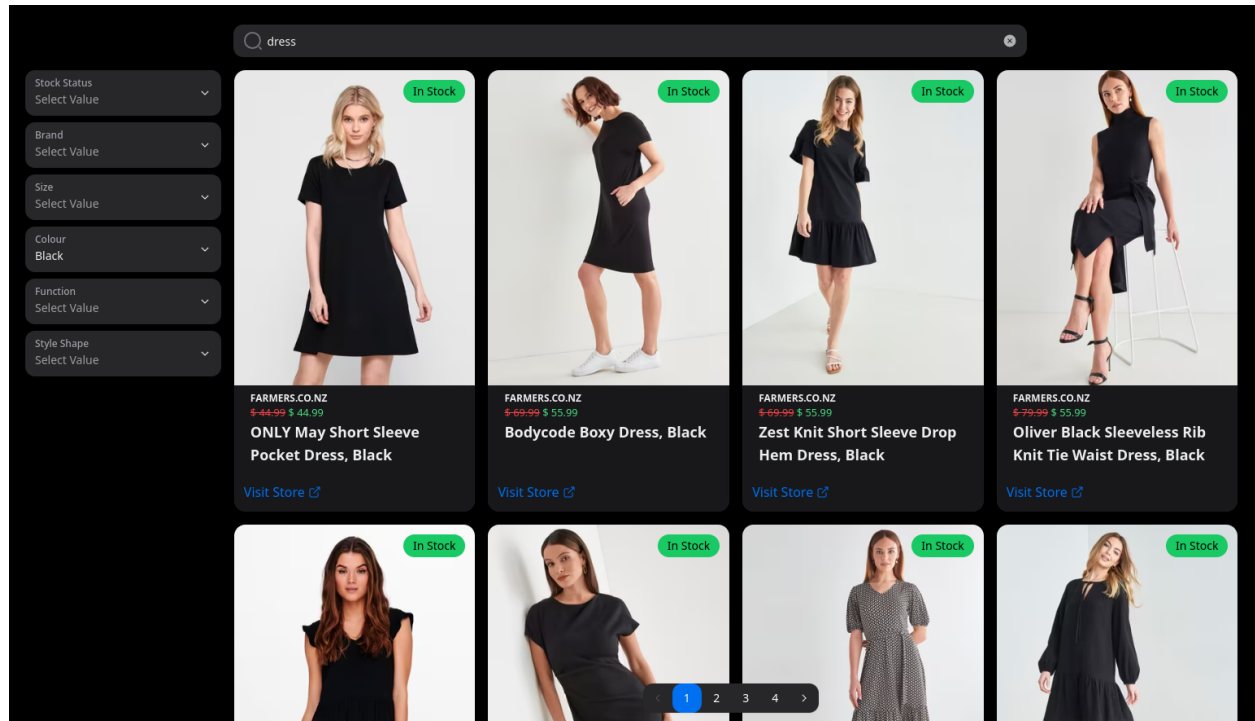


COURSE NAME

# FINAL REPORT

## Fashion Scraper Project



---

## Introduction

I thoroughly enjoy shopping, especially online. During my online shopping experiences, I frequently needed help with comparing products across various websites. The repetitive task of typing the same query and applying similar filters across different sites became cumbersome. Additionally, comparing prices was a hassle due to the distinct layouts and structures of each website. To streamline this process and make online shopping more efficient, I embarked on a project to automate these tasks. By scraping relevant data, I aimed to create a unified user interface that would allow me to search all these sites seamlessly using a single search.

As such, my project is a web-based Fashion Search Aggregation System that enables users to search for fashion-related products across multiple fashion webshops. The system provides a simple user interface where users can input search queries (e.g., "long sleeve") and receive aggregated results from various fashion webshops. For this project, only one webshop is integrated for demo purposes.

The program/scraping scripts were developed using Python and include a database for storing scraped data.

Through this project, I enhanced my programming skills and gained a better understanding of the development process, including web scraping, building a simple web GUI, implementing the necessary functionality, integrating it with a database, and testing and refining the program.

## Core Features

### Web Scraping script support

I have implemented a web scraping script architecture that allows us to add new fashion webshops in the future easily. Each webshop would be a separate script that handles the scraping logic for that specific site. This approach ensures that we can continuously expand the number of supported webshops. For the initial implementation, I focused on adding a webshop in New Zealand (<https://www.farmers.co.nz/women/fashion/dresses>).

---

The system uses Python and the BeautifulSoup package to perform web scraping for each webshop. The scraping process extracts product details, including product names, prices, and other features.

## Simple User Interface

The user interface is minimalistic and user-friendly, allowing users to input their search queries and filters.

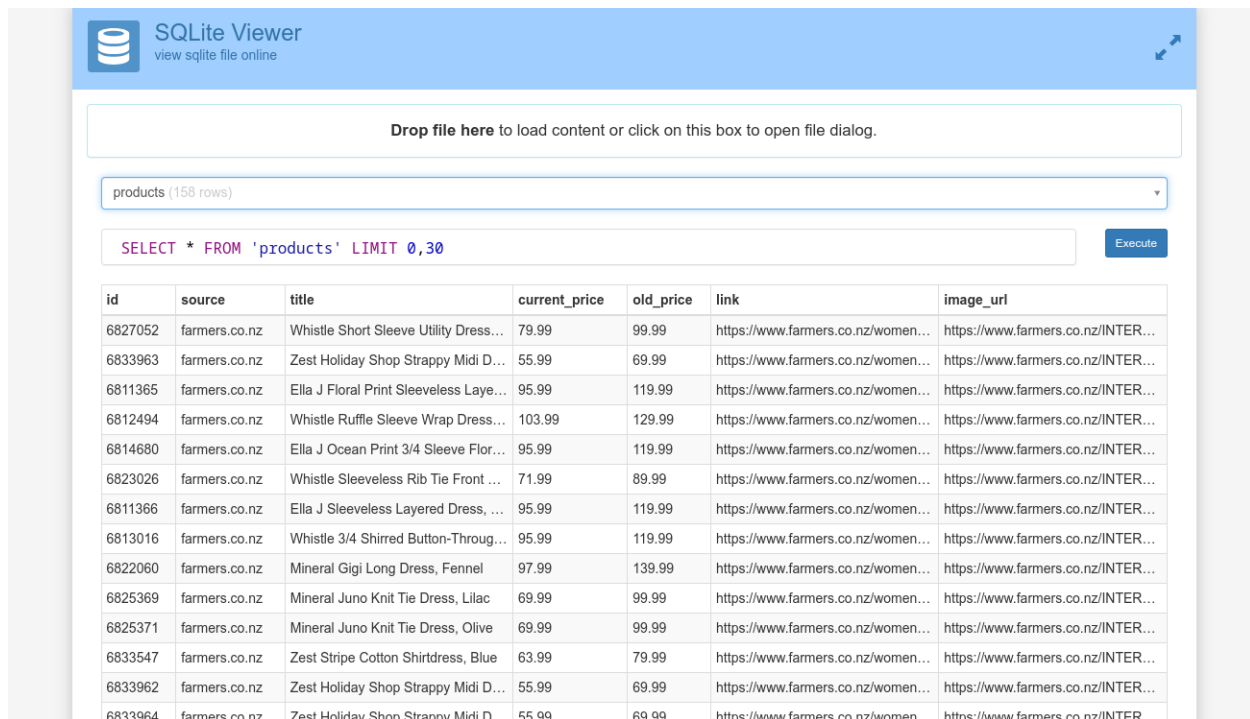
I used **React** along with a **Flask** backend to develop the web application and used Next UI (<https://nextui.org/>) for the UI components.

## Storing the Scraped Data

To improve system performance and reduce redundant scraping, a caching mechanism has been implemented that stores the scraped products.

For now, this uses a simple **SQLite database** which is updated when the script is run.

Following is the products table from the SQLite database after some products are scraped.



The screenshot shows the SQLite Viewer web application. At the top, there's a header with the SQLite logo and the text 'SQLite Viewer view sqlite file online'. Below the header is a large box with the text 'Drop file here to load content or click on this box to open file dialog.' Underneath this is a dropdown menu showing 'products (158 rows)'. Below the dropdown is a text input field containing the SQL query 'SELECT \* FROM 'products' LIMIT 0,30'. To the right of the input field is a blue button labeled 'Execute'. Below the input field is a table with 7 columns: id, source, title, current\_price, old\_price, link, and image\_url. The table contains 15 rows of data, each representing a product scraped from farmers.co.nz.

id	source	title	current_price	old_price	link	image_url
6827052	farmers.co.nz	Whistle Short Sleeve Utility Dress...	79.99	99.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6833963	farmers.co.nz	Zest Holiday Shop Strappy Midi D...	55.99	69.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6811365	farmers.co.nz	Ella J Floral Print Sleeveless Laye...	95.99	119.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6812494	farmers.co.nz	Whistle Ruffle Sleeve Wrap Dress...	103.99	129.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6814680	farmers.co.nz	Ella J Ocean Print 3/4 Sleeve Flor...	95.99	119.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6823026	farmers.co.nz	Whistle Sleeveless Rib Tie Front ...	71.99	89.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6811366	farmers.co.nz	Ella J Sleeveless Layered Dress, ...	95.99	119.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6813016	farmers.co.nz	Whistle 3/4 Shirred Button-Throug...	95.99	119.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6822060	farmers.co.nz	Mineral Gigi Long Dress, Fennel	97.99	139.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6825369	farmers.co.nz	Mineral Juno Knit Tie Dress, Lilac	69.99	99.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6825371	farmers.co.nz	Mineral Juno Knit Tie Dress, Olive	69.99	99.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6833547	farmers.co.nz	Zest Stripe Cotton Shirdress, Blue	63.99	79.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6833962	farmers.co.nz	Zest Holiday Shop Strappy Midi D...	55.99	69.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...
6833964	farmers.co.nz	Zest Holiday Shop Strappy Midi D...	55.99	69.99	https://www.farmers.co.nz/women...	https://www.farmers.co.nz/INTER...

---

## Challenges

Due to the nature of the system, I faced several challenges. Several such challenges are outlined below.

### Webshop Structure Changes

One potential challenge was that fashion webshops could update their site structure or employ measures to prevent web scraping. This was not addressed during the development phase, but in a possible product, webshop sites may need to be monitored to identify changes and disable/update the plugins.

### Real-time Scraping Efficiency

Real-time web scraping puts a strain on system resources, especially when dealing with multiple webshops concurrently. My initial plan was this, but I later planned to switch to running the script manually.

So, for the initial implementation, the focus was on a single webshop, and the script would be run on demand manually.

### Data Aggregation and Duplicate Removal

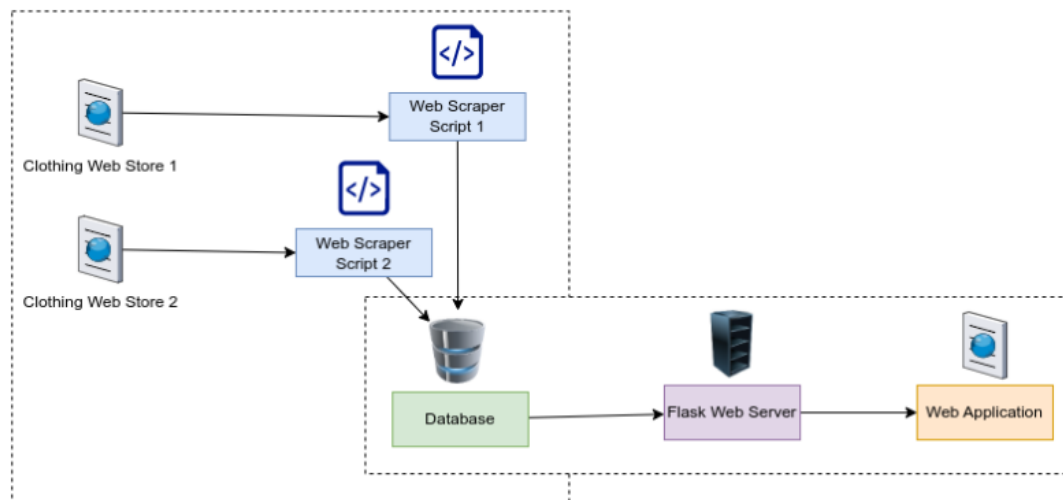
When aggregating data from multiple webshops, duplicated items were removed, although identifying duplications via code was challenging. This was not addressed in the initial development, but in a potential product, the system may need to employ AI to identify duplicates.

---

## Technical Details

Since this is an e-commerce-related scraping project, there aren't many domain-specific details to specify. The project simply browses through a website catalogue while applying different filters to index each item and store it in a database. The web application is simply a web GUI for the stored data which allows unified filtering and searching.

The following shows how every component is interconnected.



Development is done using,

- Web Scraping - Python script (code in `sync_db.py`) that uses the BeautifulSoup package (<https://pypi.org/project/beautifulsoup4/>)
- Database - Using SQLite database which is connected to other Python scripts (code in `database.py`) using SQL alchemy package (<https://www.sqlalchemy.org/>)
- Backend Web Server - Using Python (code in `api.py`) that uses Flask (<https://flask.palletsprojects.com/>)
- Web UI - Using React (code in `src/App.jsx`) and Next UI component library (<https://nextui.org/>)

---

As mentioned, only the <https://www.farmers.co.nz/women/fashion/dresses> web shop is scraped. (code in scraper.py) When sync\_db.py is run, it will create the database file file.db in the same folder. This is then served in an API by the Flask web server, which is nicely visualized in the React application.

---

# Running the project

## Initial Setup

These must be done once to set up the project.

1. Download the Python installer from Python's official website. Run the installer, ensuring you check the box that says "Add Python x.x to PATH" during installation.  
(<https://www.python.org/downloads/>)
2. Install Node.js and npm. Download the Node.js installer from the Node.js official website. Run the installer and follow the installation steps.  
(<https://nodejs.org/en/download>)
3. Open the 'scraper' folder in the command line (cmd). Run the following commands:  

```
python -m venv .venv
.venv\Scripts\activate.bat
pip install -r requirements.txt
```
4. Open the 'frontend' folder in the command line. Run the following commands:  

```
npm install --global yarn
yarn
```

## Running the web scraper

1. Open the 'scraper' folder in the command line.
2. Run the following commands to scrape. The second command will take some time to run, wait till it is finished. **If you want to use a sample database instead of scraping (which can take some time), simply rename the 'file.sample.db' file to 'file.db'.**  

```
.venv\Scripts\activate.bat
python sync_db.py
```

---

## Running the web app

1. After the web scraper has finished, check if a new file named 'file.db' is present in the 'scraper' folder.
2. Open the 'scraper' folder in the command line.
3. Run the following commands.

```
.venv\Scripts\activate.bat  
flask --app api run
```

4. This will run the server. Keep the PowerShell window open; do not close it as it's running the server.
5. Open the 'frontend' folder in the command line.
6. Run the following commands.

```
yarn dev
```

7. After some time, a URL (e.g., <http://localhost:5174/>) will be shown. Open your web browser and go to this URL to access the web app.