Sydney Events Scraper & Dashboard — Project Report

Name: Akshat Tanwar
Program: B.Tech
Assignment: Event Scraping & Event Management Platform

Objective

The objective of this project was to develop a full-stack web application that automatically discovers and displays public events in Sydney, Australia. The system was required to scrape event data from public sources, store it in a database, present it in a clean user interface, and provide an admin dashboard with filtering and import capabilities. A key requirement was to keep the event database continuously updated by detecting new, modified, and inactive events.

System Overview

The platform implements a complete automated pipeline:

Scrape → Store → Display → Review → Import → Auto-Update

Public users can browse events through a minimal Bootstrap-based interface, while administrators can review and manage events through a dashboard. The system is designed to run autonomously using scheduled scraping.

Technology Stack

Frontend

- HTML5
- Bootstrap 5
- Vanilla JavaScript

Backend

- Node.js
- Express.js
- MongoDB with Mongoose

Automation & Scraping

- Puppeteer (dynamic scraping)
- node-cron (scheduled execution)

Version Control

- Git & GitHub

Key Features

1. Automated Event Scraping
Events are scraped from Eventbrite Sydney using Puppeteer. The scraper extracts key fields such as title, date/time, venue, description, image URL, source, and original event link. Duplicate prevention is implemented using unique URLs.

2. Automatic Status Detection
Each event is automatically tagged as **new**, **updated**, **inactive**, or **imported**. After every scrape, events not refreshed within a cutoff window are marked inactive, ensuring database freshness.

3. Public Event Listing UI
A clean, minimal Bootstrap interface displays event cards with essential details and a **GET TICKETS** call-to-action. Users must submit email consent before being redirected, and the capture is stored in MongoDB.

4. Admin Dashboard
The dashboard provides keyword search, city and date filters, table view, status badges, preview panel, and an **Import** action that updates event metadata (`importedAt`, `importedBy`, notes).

5. Scheduled Auto Updates
Using node-cron, the scraper runs periodically to detect newly published events, update changed records, and deactivate expired ones. This ensures the platform remains continuously up to date.

Database Design

MongoDB is used with a modular Event schema for scalability and maintainability. The design supports status tracking, import metadata, and user email captures.

Challenges & Solutions

Key challenges included static file serving issues, port conflicts on macOS, cross-origin fetch problems when using `file://`, and Puppeteer selector tuning. These were resolved through proper Express static configuration, correct port management, and robust scraping logic.

Conclusion

The project successfully demonstrates an end-to-end automated event discovery platform using open-source technologies. It highlights practical skills in web scraping, REST API development, database management, frontend rendering, and scheduled automation. The system is scalable and can be extended to multi-city support or personalized recommendations in the future.

GitHub Repository: *(add your repo link here)*