📘 PROJECT REPORT

Sydney Events Aggregation Platform (MERN Stack)

---

1. Introduction

The Sydney Events Aggregation Platform is a full-stack MERN application developed to automatically collect, manage, and display events from public event websites for Sydney, Australia.

The primary goal of this project was to build a scalable system that:

- Scrapes event data automatically
- Detects event lifecycle changes (new, updated, inactive)
- Displays events in a minimal, user-friendly UI
- Captures user emails before ticket redirection
- Provides a Google OAuth-protected admin dashboard
- Allows event review and import tracking

This project demonstrates end-to-end development capabilities including scraping, backend API design, authentication, database modeling, and frontend interface development.

---

2. Objectives

The main objectives of this project were:

1. Automatically scrape Sydney events from public websites.
2. Store structured event data in MongoDB.
3. Detect new, updated, and inactive events.
4. Create a clean and modern event listing website.
5. Implement email capture with consent before redirecting users.
6. Integrate Google OAuth authentication.
7. Develop a dashboard with filtering and import workflow.
8. Demonstrate full scrape → store → display → review → import lifecycle.

---

3. Technology Stack

Frontend

- React (Vite)
- Axios
- React Router DOM
- CSS (inline styling)

Backend

- Node.js
- Express.js
- Mongoose
- Passport.js (Google OAuth)
- JWT Authentication
- Axios + Cheerio (Web Scraping)

Database

- MongoDB Atlas (Cloud)

---

4. System Architecture

The system follows a standard MERN architecture:

Scraper → Backend API → MongoDB → React Frontend → Admin Dashboard

Data Flow:

1. Scraper collects event data.
2. Data is processed and stored in MongoDB.
3. API serves events to frontend.
4. Public UI displays events.
5. Admin reviews events in dashboard.
6. Admin imports selected events.
7. Status tags update accordingly.

---

5. Event Scraping & Data Processing

The scraper extracts event data from public Sydney event listings.

Data Collected:

- Title
- Date & Time

- Venue Name
- City
- Description
- Category
- Image URL
- Source Website
- Original Event URL
- Last Scraped Time
- Status Tag

Lifecycle Detection

The system automatically detects:

1️⃣ **New Events**

If event URL does not exist in database → status set to new.

2️⃣ **Updated Events**

If event exists but details changed → status set to updated.

3️⃣ **Inactive Events**

If event no longer appears in latest scrape → status set to inactive.

4️⃣ **Imported Events**

If admin manually imports → status set to imported.

This ensures complete lifecycle management.

---

6. Database Design

Event Schema

```
{
  title: String,
  dateTime: String,
  venueName: String,
  city: String,
  description: String,
  category: String,
  imageUrl: String,
  sourceWebsite: String,
  originalUrl: String,
  status: String,
  lastScrapedAt: Date,
```

```
  importedAt: Date,
  importedBy: String
}
```

Subscription Collection

```
{
  email: String,
  consent: Boolean,
  eventId: ObjectId,
  createdAt: Date
}
```

---

7. Public Website Implementation

The public interface displays active events in a clean grid layout.

Each event card shows:

- Event name
- City
- Source
- GET TICKETS button

Ticket Flow:

1. User clicks GET TICKETS.
2. Modal requests:
    - Email address
    - Consent checkbox
3. Email is saved in database.
4. User redirected to original event page.

This ensures lead capture before traffic redirection.

---

8. Google OAuth Authentication

Google OAuth was implemented using Passport.js.

Authentication Flow:

1. User clicks Admin Login.
2. Redirected to Google login.
3. Google returns user profile.

4. Backend generates JWT.
5. Token sent to frontend.
6. Protected routes verify JWT.

Only authenticated users can access dashboard features.

---

9. Admin Dashboard Features

The dashboard provides event management functionality.

1️⃣ Table View

Displays:

- Title
- Status
- Source
- Imported By
- Imported At
- Import Button

2️⃣ Status Tags

Color-coded badges:

- Green → New
- Orange → Updated
- Red → Inactive
- Blue → Imported

3️⃣ Filters

- Search filter (by title)
- City filter (default: Sydney)
- Real-time filtering

4️⃣ Analytics Summary

Dashboard header displays:

- Total events
- New events
- Updated events
- Imported events

- Inactive events

5️⃣ Preview Panel

Clicking an event row shows:

- Full description
- Venue details
- City
- Source
- Direct link

6️⃣ Import Workflow

Admin can import an event:

- Status changes to imported
- importedAt timestamp stored
- importedBy (admin email) stored
- Import button disabled afterward

---

10. Security Implementation

- JWT-based authentication
- Protected API routes
- Environment variables stored in .env
- .gitignore prevents sensitive data exposure
- CORS properly configured
- MongoDB Atlas secured with IP access control

---

11. Challenges Faced

1. Handling CORS issues between frontend and backend.
2. Managing JWT token persistence after login.
3. Implementing inactive event detection.
4. Debugging OAuth callback flows.
5. Managing real-time UI updates after import.

Each challenge was resolved through systematic debugging and backend configuration updates.

---

12. Results Achieved

The system successfully demonstrates:

- Automated event aggregation
- Lifecycle detection logic
- Lead capture integration
- Authentication & authorization
- Admin workflow management
- Real-time status tracking
- Clean and responsive UI

All assignment requirements were fulfilled.

---

13. Future Enhancements

- Multi-city expansion
- Pagination
- Advanced date filtering
- Role-based access control
- Production deployment
- Improved UI styling
- Scheduled scraping automation

---

14. Conclusion

The Sydney Events Aggregation Platform is a complete end-to-end MERN application that demonstrates:

- Backend automation
- Database modeling
- Authentication systems
- REST API architecture
- Frontend UI design
- Lifecycle state management
- Secure admin workflows

This project reflects strong full-stack development capability and practical implementation of real-world system design.

---

15. Author

Khwahish
MERN Stack Developer
Backend Systems | Web Scraping | Authentication & API Design