Web Scraping and Deployment of an Event Listing Application

1. Introduction

This is a Node.js and Express-based web scraper that fetches events from Meetup and serves them via an API. The frontend displays the events and includes a booking modal that asks for an email and OTP before proceeding.

- Brief overview of the project:
    - Scraping event data from Meetup using Puppeteer.
    - Storing data in a MongoDB database.
    - Displaying events in a React frontend.
    - Deploying backend (Node.js) on Railway and frontend (React) on Vercel.

2. Functionality

- **Scraping**: Puppeteer was used to scrape event data from the Meetup website and store it in a MongoDB database.
- **Displaying Data**: The scraped event data is retrieved from the MongoDB database using the Axios library in the React app and displayed in the form of cards.
- **Booking Feature**: Each event card includes a “Book Tickets” button. When clicked, users are prompted to enter their email, followed by OTP verification. Once verified, they are redirected to the related event's detail page.
- **Data Update**: The event data in the database is updated every 24 hours. Initially, a cron library was used for scheduled updates, but due to issues, a set interval approach was implemented to periodically fetch and update the data.

3. Web Scraping Process

- **Tool Used:** Puppeteer
- **Target Website:** Meetup event listing page
- **Scraping Steps:**
    1. Launched Puppeteer browser.
    2. Navigated to the event listing page.
    3. Extracted event details (title, date, hostedBy, link).
    4. Stored data in MongoDB after clearing old entries.

4. Frontend Development & Deployment

- **React app fetching scraped events from the backend.**
- **UI Features:**
    - Dark theme with event cards.
    - Booking system with OTP verification (dummy implementation).

- **Deployment Steps (Vercel):**
    1. Pushed React code to GitHub.
    2. Linked GitHub repo to Vercel.
    3. Faced build error: "react-icons/io5" not found.
    4. **Fixes Applied:**
        - Installed react-icons.
        - Ensured correct import format.
        - Cleared cache & reinstalled dependencies.

5. Challenges :

1. **CORS Conflict:**
    - **Challenge:** The React frontend faced CORS (Cross-Origin Resource Sharing) issues when trying to fetch data from the backend server.
    - **Solution:** The issue was resolved by configuring the backend to include the appropriate CORS headers, allowing cross-origin requests from the frontend.
2. **Data Update Scheduling**:
    - **Challenge**: Initially, the scheduled data update was set up using the cron library, but it failed to work reliably.
    - **Solution**: The data update mechanism was switched to a set interval approach, running at regular intervals to update the scraped event data every 24 hours.
3. **Event Data Formatting**:
    - **Challenge**: The event data scraped from the Meetup website often had inconsistent formats, missing fields, or unexpected data types.
    - **Solution**: Added validation and fallback values to handle missing or malformed data, ensuring consistent card rendering in the React app.

5. Improvements :

- **Email Verification with OTP**: A real email verification system can be added on the backend to send OTPs via email. This would improve security and user experience, ensuring only valid users can proceed with bookings.
- **Event Details Page**: A dedicated event details page can be created for each event selected. This page can provide more in-depth information about the event, such as location maps, speakers, and other relevant details.