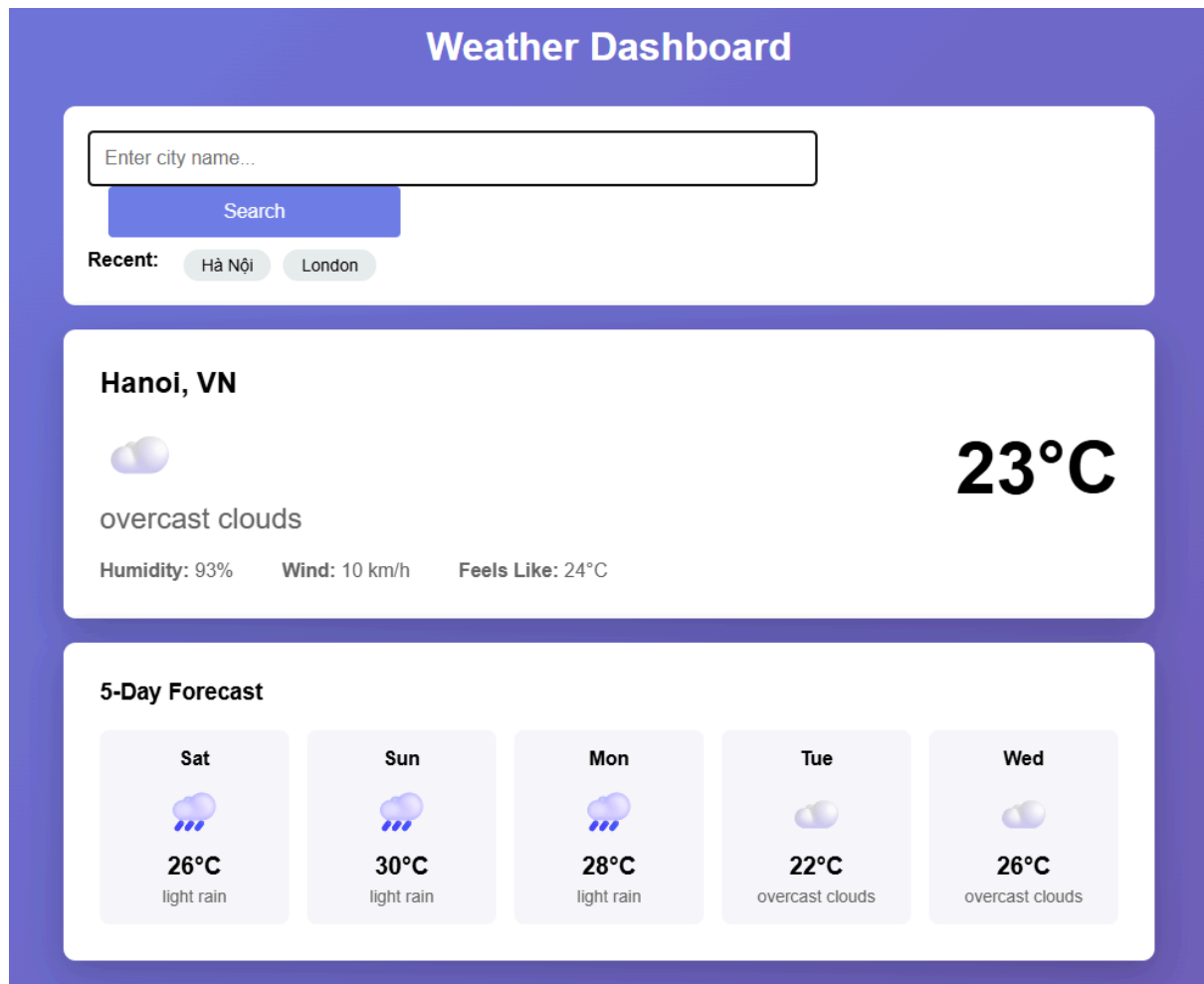


Lab 3 - Exercise 2 Report

Task 2.1: Weather Dashboard



What:

A web application that displays current weather and 5-day forecasts for any city using the OpenWeatherMap API. Users can search cities, view weather details (temperature, humidity, wind speed), and access recent searches stored locally.

Why:

Weather dashboards demonstrate real-world API integration, asynchronous JavaScript (async/await), error handling, and localStorage implementation. These skills are essential for

building data-driven applications that fetch and display external data. Weather APIs are commonly used in travel sites, mobile apps, and smart home systems.

How:

- API Integration: Used `fetch()` with OpenWeatherMap API endpoints (`/data/2.5/weather` and `/data/2.5/forecast`) to retrieve weather data
- Async Operations: Implemented `async/await` in `fetchWeather()` and `fetchForecast()` functions to handle API requests
- Data Processing: Filtered forecast data to show one entry per day (noon time) and mapped weather conditions to emoji icons
- Error Handling: Added try-catch blocks to handle 404 (city not found), 401 (invalid API key), and network errors with user-friendly messages
- Local Storage: Saved up to 5 recent searches using `localStorage.setItem()` and loaded them with `localStorage.getItem()` for quick re-searching
- UI Updates: Dynamically updated DOM with `innerHTML` to display weather cards, loading states, and error messages

Task 2.2: GitHub Repository Finder



What:

A search tool that queries GitHub's public API to find repositories by keyword, displaying results with repository details (name, description, stars, forks, language). Features include sorting options (stars/forks/updated) and pagination to load more results.

Why:

GitHub API integration teaches working with RESTful APIs, handling pagination, and processing JSON data structures. This simulates real developer tools like code search platforms, portfolio sites, and repository analytics dashboards. Understanding GitHub's API is valuable for building developer tools and automation scripts.

How:

- API Search: Used GitHub's search API (`/search/repositories`) with query parameters for search term, sort option, and pagination (`page` and `per_page`)

- Dynamic Display: Created repository cards dynamically with `createRepoCard()` function, showing owner, name, stars, forks, language, and description
- Pagination: Implemented "Load More" functionality by incrementing `currentPage` and appending new results to existing ones without replacing them
- Sorting: Added dropdown for sorting by stars, forks, or updated date, triggering new searches when changed
- Error Handling: Detected rate limiting (403 status), invalid queries (422 status), and network errors with appropriate user messages
- Number Formatting: Used `formatNumber()` to convert large numbers (e.g., 180000 → 180k, 1500000 → 1.5M) for better readability
- No Authentication: Leveraged GitHub's 60 requests/hour limit for unauthenticated users (no API key required)