Ex: 2.1
Input

```
// Fetch current weather data
async function fetchWeather(city) {
    try {
        const response = await fetch(
            `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${API_KEY}&units=metric`
        );

        if (!response.ok) {
            if (response.status === 404) {
                throw new Error(`City "${city}" not found. Please check the spelling.`);
            } else if (response.status === 401) {
                throw new Error('Invalid API key. Please check your configuration.');
            } else {
                throw new Error('Weather service is currently unavailable. Please try again later.');
            }
        }

        return await response.json();
    } catch (error) {
        if (error.message.includes('Failed to fetch')) {
            throw new Error('Network error. Please check your internet connection.');
        }
        throw error;
    }
}
```

Calls OpenWeatherMap API with city name and API key
Handles response status:
- 200: Success → returns JSON data
- 404: City doesn't exist → shows error message
- 401: Invalid API key
- Others: Service unavailable

Error handling for network errors
Uses async/await for asynchronous processing

```
// Fetch 5-day forecast data
async function fetchForecast(city) {
    try {
        const response = await fetch(
            `https://api.openweathermap.org/data/2.5/forecast?q=${city}&appid=${API_KEY}&units=metric`
        );

        if (!response.ok) {
            throw new Error('Unable to fetch forecast data.');
        }

        return await response.json();
    } catch (error) {
        throw new Error(`Failed to fetch forecast: ${error.message}`);
    }
}
```

Differences from fetchWeather:
- **Different API endpoint**: /forecast instead of /weather
- **Returns 5-day data** with 3-hour intervals
- **Simpler error handling** since city is already validated in fetchWeather

```
// Display current weather
function displayWeather(data) {
    const weatherDisplay = document.getElementById('weatherDisplay');
    const icon = weatherIcons[data.weather[0].main] || '🌤';

    weatherDisplay.innerHTML = `
        <div class="weather-card">
            <div class="city-name">${data.name}, ${data.sys.country}</div>
            <div class="weather-description">${data.weather[0].description}</div>

            <div class="current-weather">
                <div class="current-weather-left">
                    <div class="weather-icon">${icon}</div>
                    <div class="temp-display">${Math.round(data.main.temp)}°C</div>
                </div>
                <div class="current-weather-right">
                    <div class="weather-details">
                        <div class="detail-item">
                            <span>Feels Like:</span>
                            <span><strong>${Math.round(data.main.feels_like)}°C</strong></span>
                        </div>
                        <div class="detail-item">
                            <span>Humidity:</span>
                            <span><strong>${data.main.humidity}%</strong></span>
                        </div>
                        <div class="detail-item">
                            <span>Wind Speed:</span>
                            <span><strong>${data.wind.speed} m/s</strong></span>
                        </div>
                        <div class="detail-item">
                            <span>Pressure:</span>
                            <span><strong>${data.main.pressure} hPa</strong></span>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    `;
```

Receives data object from API response
Maps weather condition → emoji icon
Creates HTML template with: city năm % country, weather description, main temperature, details (feels like, humidity, wind speed, pressure)
Responsive layout: 2-column on desktop, stacked on mobile

```
function displayForecast(data) {
        const forecastDisplay = document.getElementById('forecastDisplay');

        // Filter to get one forecast per day (around noon)
        const dailyForecasts = [];
        const processedDays = new Set();

        data.list.forEach(item => {
            const date = new Date(item.dt * 1000);
            const day = date.toDateString();

            // Get forecast around noon (12:00) or the first available for each day
            if (!processedDays.has(day)) {
```

```
                processedDays.add(day);
                dailyForecasts.push(item);
            }
        });

        // Take only 5 days
        const fiveDayForecast = dailyForecasts.slice(0, 5);

        let forecastHTML = `
            <div class="weather-card">
                <div class="section-title">5-Day Forecast</div>
                <div class="forecast-grid">
        `;

        fiveDayForecast.forEach(day => {
            const date = new Date(day.dt * 1000);
            const dayName = date.toLocaleDateString('en-US', { weekday: 'short' });
            const fullDate = date.toLocaleDateString('en-US', { month: 'short', day:
'numeric' });
            const icon = weatherIcons[day.weather[0].main] || '   ';

            forecastHTML += `
                <div class="forecast-item">
                    <div class="forecast-
date">${dayName}<br><small>${fullDate}</small></div>
                    <div class="forecast-icon">${icon}</div>
                    <div class="forecast-temp">${Math.round(day.main.temp)}°C</div>
                    <div style="font-size: 14px; color: #718096; margin-top:
8px;">${day.weather[0].description}</div>
                </div>
            `;
        });

        forecastHTML += `
                </div>
            </div>
        `;

        forecastDisplay.innerHTML = forecastHTML;
    }
```

Filter data: API returns forecast every 3 hours → take only 1 sample per day
Use Set() to track processed days
Grid layout with 5 forecast items
Display: Day, icon, temperature, description

```javascript
// Main search function
async function searchWeather() {
    const cityInput = document.getElementById('cityInput');
    const city = cityInput.value.trim();

    if (!city) {
        showError('Please enter a city name');
        return;
    }

    // Show loading state
    showLoading();
    clearError();

    try {
        // Fetch both current weather and forecast simultaneously
        const [weatherData, forecastData] = await Promise.all([
            fetchWeather(city),
            fetchForecast(city)
        ]);

        // Display data
        displayWeather(weatherData);
        displayForecast(forecastData);

        // Save to recent searches
        saveRecentSearch(city);

    } catch (error) {
        showError(error.message);
        document.getElementById('weatherDisplay').innerHTML = '';
        document.getElementById('forecastDisplay').innerHTML = '';
    }
}
```

Input validation: Check if city is not empty
Loading state: Show spinner while fetching
Parallel requests: Use Promise.all() to fetch current and forecast data simultaneously
Error handling: Catch and display errors
State persistence: Save search to localStorage

```
// Save recent search to localStorage
function saveRecentSearch(city) {
    let recentSearches = JSON.parse(localStorage.getItem('recentSearches')) || [];

    // Remove if already exists (case insensitive)
    recentSearches = recentSearches.filter(item =>
        item.toLowerCase() !== city.toLowerCase()
    );

    // Add to beginning
    recentSearches.unshift(city);

    // Keep only 5 most recent
    if (recentSearches.length > 5) {
        recentSearches = recentSearches.slice(0, 5);
    }

    localStorage.setItem('recentSearches', JSON.stringify(recentSearches));
    loadRecentSearches();
}
```

Read from localStorage → parse JSON
Deduplication: Remove duplicates (case insensitive)
FIFO logic: Add new items to beginning, limit to 5 items
Persist: Save back to localStorage
Refresh UI: Call `loadRecentSearches()` to update display

```
// Load and display recent searches
function loadRecentSearches() {
    const recentSearches = JSON.parse(localStorage.getItem('recentSearches')) || [];
    const recentSearchesElement = document.getElementById('recentSearches');

    if (recentSearches.length === 0) {
        recentSearchesElement.innerHTML = '';
        return;
    }

    let html = '<span style="color: #718096; font-weight: 500;">Recent searches:</span>';
    recentSearches.forEach(city => {
        html += `<div class="recent-city" onclick="searchRecentCity('${city}')">${city}</div>`;
    });

    recentSearchesElement.innerHTML = html;
}
```

Read data from localStorage
Create clickable buttons for each recent city
Event handling: Click → auto-fill and search

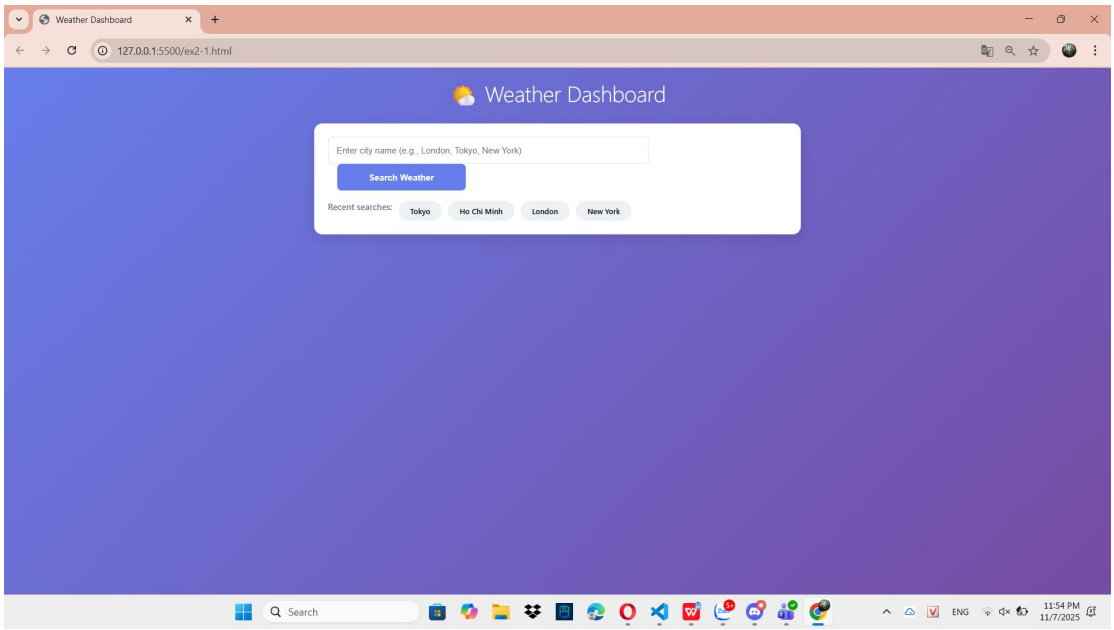
```
// Show error message
function showError(message) {
    const errorElement = document.getElementById('errorMessage');
    errorElement.innerHTML = `<div class="error">${message}</div>`;
}

// Clear error message
function clearError() {
    document.getElementById('errorMessage').innerHTML = '';
}
```

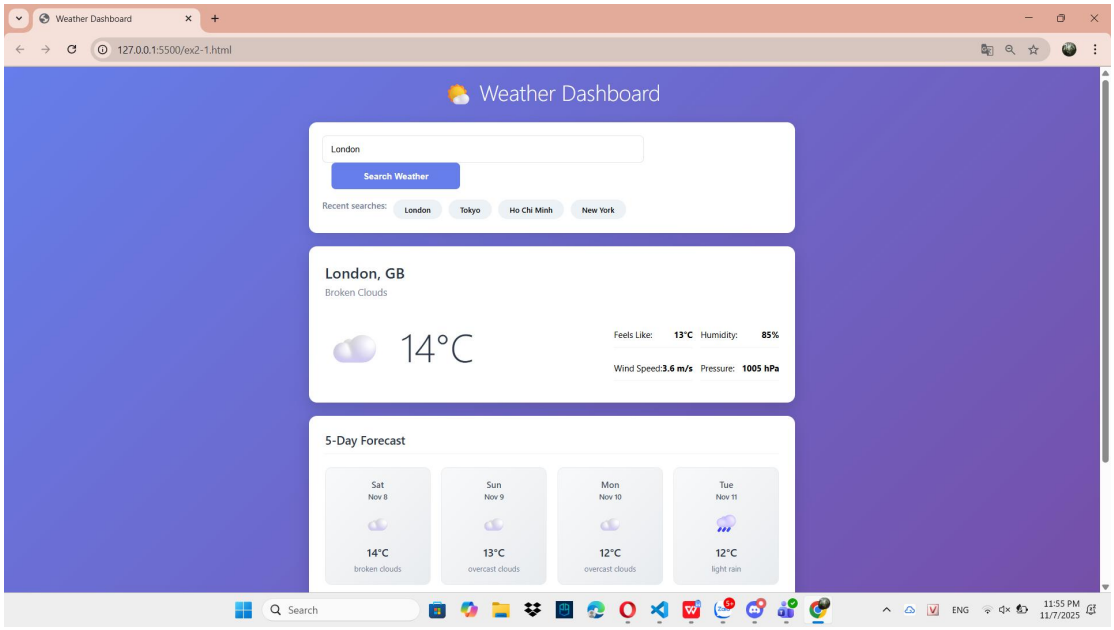
Error display: Show message in styled error container
Auto-clear: Clear error before new search

```
// Show loading state
function showLoading() {
    document.getElementById('weatherDisplay').innerHTML = `
        <div class="loading">
            <div class="spinner"></div>
            <p>Loading weather data for ${document.getElementById('cityInput').value}...</p>
        </div>
    `;
    document.getElementById('forecastDisplay').innerHTML = '';
}
```

Visual feedback: Show spinner animation
Visual feedback: Show spinner animation
Clear previous data: Clear old forecast display

OUTPUT





EX: 2.2

INPUT

```
// Search repositories from GitHub API
async function searchRepositories(query, sort = 'stars', page = 1) {
    try {
        const response = await fetch(
            `https://api.github.com/search/repositories?q=${encodeURIComponent(query)}&sort=${sort}&page=${page}&per_page=10`
        );

        if (!response.ok) {
            if (response.status === 403) {
                throw new Error('GitHub API rate limit exceeded. Please wait a moment and try again.');
            } else if (response.status === 422) {
                throw new Error('Invalid search query. Please try a different search term.');
            } else {
                throw new Error(`GitHub API error: ${response.status}`);
            }
        }

        const data = await response.json();
        return data;

    } catch (error) {
        throw error;
    }
}
```

Constructs API URL with query parameters:

- q: Search query (URL encoded)
- sort: Sorting method (stars, forks, updated)
- page: Pagination page number
- per_page=10: Limits results to 10 per page

Handles HTTP errors:

- 403: Rate limit exceeded (60 requests/hour for unauthenticated)
- 422: Invalid search query

Others: Generic GitHub API error

Returns parsed JSON containing repository data

```
// Display repositories
function displayRepositories(repos, append = false) {
    const repoList = document.getElementById('repoList');

    if (!append) {
        repoList.innerHTML = '';
    }

    if (repos.length === 0) {
        if (!append) {
            repoList.innerHTML = `
                <div class="empty-state">
                    <h3>No repositories found</h3>
                    <p>Try searching for something else</p>
                </div>
            `;
        }
        return;
    }

    repos.forEach(repo => {
        repoList.appendChild(createRepoCard(repo));
    });
}
```

Manages DOM updates:

- Append = false: Clear existing results (new search)
- Apeend = true: Add to existing results (load more)

Handles empty results with user-friendly message

Iterates through repositories and creates cards for each

```
// Create repository card HTML
    function createRepoCard(repo) {
        const card = document.createElement('div');
        card.className = 'repo-card';

        card.innerHTML = `
            <a href="${repo.html_url}" target="_blank" class="repo-name">
                ${repo.full_name}
            </a>

            ${repo.description ? `
                <div class="repo-description">
                    ${repo.description}
                </div>
            ` : ''}

            <div class="repo-meta">
                <div class="meta-item">
                    ⭐ ${formatNumber(repo.stargazers_count)}
                </div>
                <div class="meta-item">
                    ${formatNumber(repo.forks_count)}
                </div>
```

```
                ${repo.language ? `
                    <div class="meta-item">
                        <span class="language-badge">${repo.language}</span>
                    </div>
                ` : ''}
                <div class="meta-item">
                        Updated ${formatDate(repo.updated_at)}
                </div>
            </div>

            <div class="owner-info">
                <img src="${repo.owner.avatar_url}" alt="${repo.owner.login}"
class="owner-avatar">
                <span>by ${repo.owner.login}</span>
            </div>
        `;

        return card;
    }
```

Repository name: Clickable link to GitHub
Description: Conditional display (some repos have no description)
Metadata: Stars, forks, language, last updated
Owner info: Avatar and username
Conditional rendering: Only shows language if available

```
// Main search function
    async function performSearch() {
        const searchInput = document.getElementById('searchInput');
        const query = searchInput.value.trim();
        const sort = document.getElementById('sortSelect').value;

        if (!query) {
            showError('Please enter a search term');
            return;
        }

        // Reset for new search
        currentPage = 1;
        currentQuery = query;
        currentSort = sort;
        isLoading = true;

        showLoading();
        clearError();

        try {
            const data = await searchRepositories(query, sort, currentPage);
            totalResults = data.total_count;
```

```
            displayResultsInfo(data.total_count);
            displayRepositories(data.items);
            setupLoadMore(data.total_count);

        } catch (error) {
            showError(error.message);
            document.getElementById('repoList').innerHTML = '';
            document.getElementById('loadMoreContainer').innerHTML = '';
        } finally {
            isLoading = false;
        }
    }
```

State Management:

- Global variables:

+ currenPage: Tracks current page for pagination
+ currentQuery: Stores current search term
+ currentSort: Stores current sort method
+ totalResults: Total number of repositories found
+ isLoading: Prevents multiple simultaneous requests

Workflow:

- Input validation - Check if query is not empty
- Reset state - Prepare for new search
- Show loading - User feedback
- API call - Fetch repositiries
- Update UI - Display result and pagination
- Error handing - Clear displays on error

```
// Load more results
async function loadMore() {
    if (isLoading) return;

    currentPage++;
    isLoading = true;

    const loadMoreButton = document.getElementById('loadMoreButton');
    loadMoreButton.innerHTML = 'Loading...';
    loadMoreButton.disabled = true;

    try {
        const data = await searchRepositories(currentQuery, currentSort, currentPage);
        displayRepositories(data.items, true);
        setupLoadMore(totalResults);

    } catch (error) {
        showError(error.message);
        currentPage--; // Revert page on error
    } finally {
        isLoading = false;
    }
}
```

Pagination Features:

- Incremental loading - Adds to existing results
- Button state management - Disabled during loading
- Error recovery - Reverts page number on failure
- Loading indicator - Button text changes during fetch

```
// Set up load more button
function setupLoadMore(totalCount) {
    const loadMoreContainer = document.getElementById('loadMoreContainer');
    const displayedCount = currentPage * 10;

    if (displayedCount < totalCount) {
        loadMoreContainer.innerHTML = `
            <div class="load-more">
                <button id="loadMoreButton" onclick="loadMore()">
                    Load More Results (${displayedCount}/${totalCount})
                </button>
            </div>
        `;
    } else {
        loadMoreContainer.innerHTML = '';
    }
}
```

Pagination Logic:
- **Calculates displayed count** - `currentPage * 10` (10 items per page)
- **Shows/hides button** - Only shows when more results available
- **Progress indicator** - Shows current position (e.g., "20/1000")

```
// Format large numbers
function formatNumber(num) {
    if (num >= 1000000) {
        return (num / 1000000).toFixed(1) + 'M';
    } else if (num >= 1000) {
        return (num / 1000).toFixed(1) + 'k';
    }
    return num.toString();
}
```

Formats large numbers:
1,500,000 → "1.5M"
25,000 → "25.0k"
999 → "999"

```
// Format date
function formatDate(dateString) {
    const date = new Date(dateString);
    const now = new Date();
    const diffTime = Math.abs(now - date);
    const diffDays = Math.ceil(diffTime / (1000 * 60 * 60 * 24));

    if (diffDays === 1) return 'today';
    if (diffDays <= 7) return `${diffDays} days ago`;
    if (diffDays <= 30) return `${Math.ceil(diffDays / 7)} weeks ago`;
    return `${Math.ceil(diffDays / 30)} months ago`;
}
```

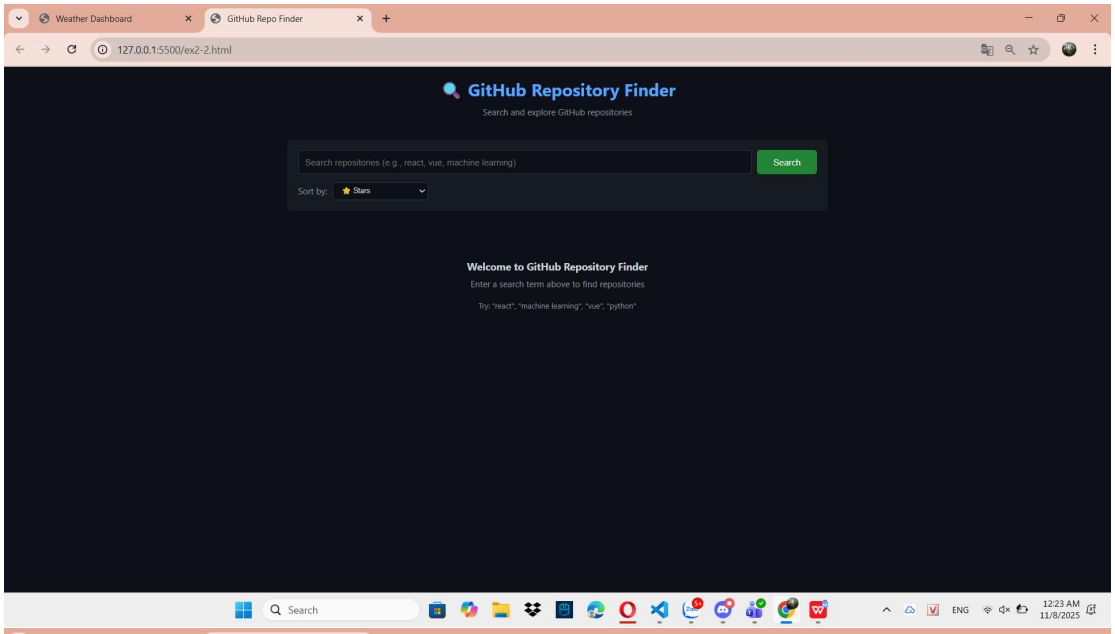
Creates relative time strings: "today", "2 days ago", "3 weeks ago", "5 months ago"

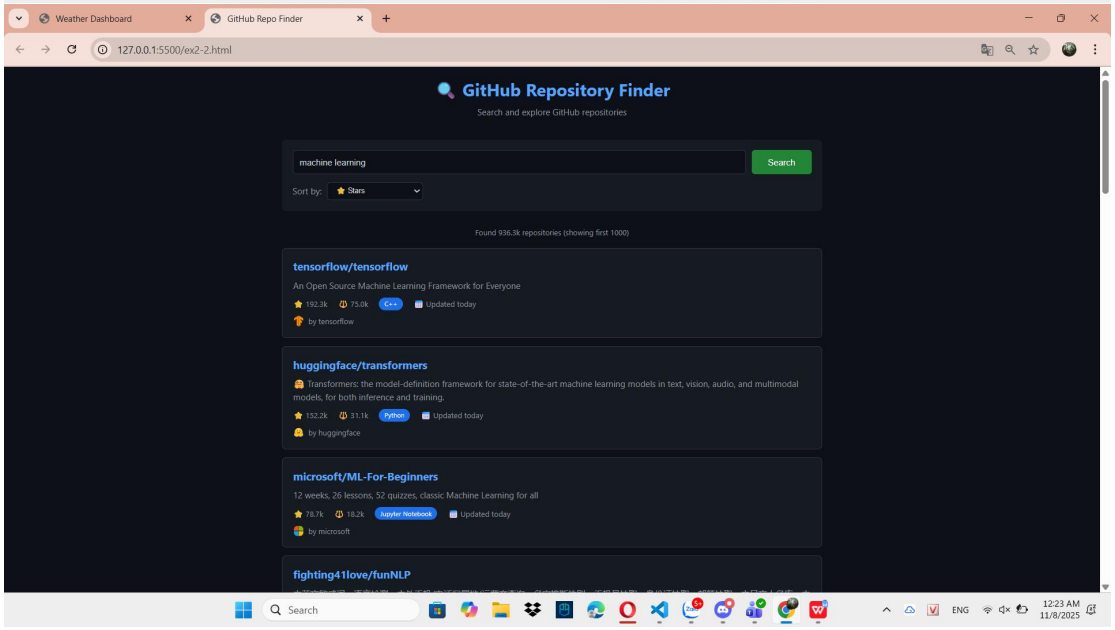
```
// Show loading state
function showLoading() {
    const repoList = document.getElementById('repoList');
    repoList.innerHTML = `
        <div class="loading">
            <div class="spinner"></div>
            <p>Searching GitHub repositories...</p>
        </div>
    `;
}

// Show error message
function showError(message) {
    const errorElement = document.getElementById('errorMessage');
    errorElement.innerHTML = `<div class="error">${message}</div>`;
}

// Clear error message
function clearError() {
    document.getElementById('errorMessage').innerHTML = '';
}
```

User Experience Features:

- **Loading spinner** - Visual feedback during API calls
- **Error messages** - Clear, actionable error information
- **Empty states** - Helpful messages when no results found
- **Button states** - Disabled during loading to prevent duplicate requests

OUTPUT


Weather Dashboard
GitHub Repo Finder
127.0.0.1:5500/ex2-2.html
GitHub Repository Finder
Search and explore GitHub repositories
Search repositories (e.g., react, vue, machine learning)
Search
Sort by:
Stars
Welcome to GitHub Repository Finder
Enter a search term above to find repositories
Try: "react", "machine learning", "vue", "python"
Search
ENG
12:23 AM
11/8/2025


Weather Dashboard
GitHub Repo Finder
127.0.0.1:5500/ex2-2.html
GitHub Repository Finder
Search and explore GitHub repositories
machine learning
Search
Sort by:
Stars
Found 936.3k repositories (showing first 1000)
tensorflow/tensorflow
An Open Source Machine Learning Framework for Everyone
192.3k
75.0k
C++
Updated today
by tensorflow
huggingface/transformers
Transformers: the model-definition framework for state-of-the-art machine learning models in text, vision, audio, and multimodal models, for both inference and training.
152.2k
31.1k
Python
Updated today
by huggingface
microsoft/ML-For-Beginners
12 weeks, 26 lessons, 52 quizzes, classic Machine Learning for all
78.7k
18.2k
Jupyter Notebook
Updated today
by microsoft
fighting41love/funNLP
Search
ENG
12:23 AM
11/8/2025