REPORT ASSIGNMENT LAB 2

Student Name: Nguyễn Tiến Anh

Student ID: ITITDK22128

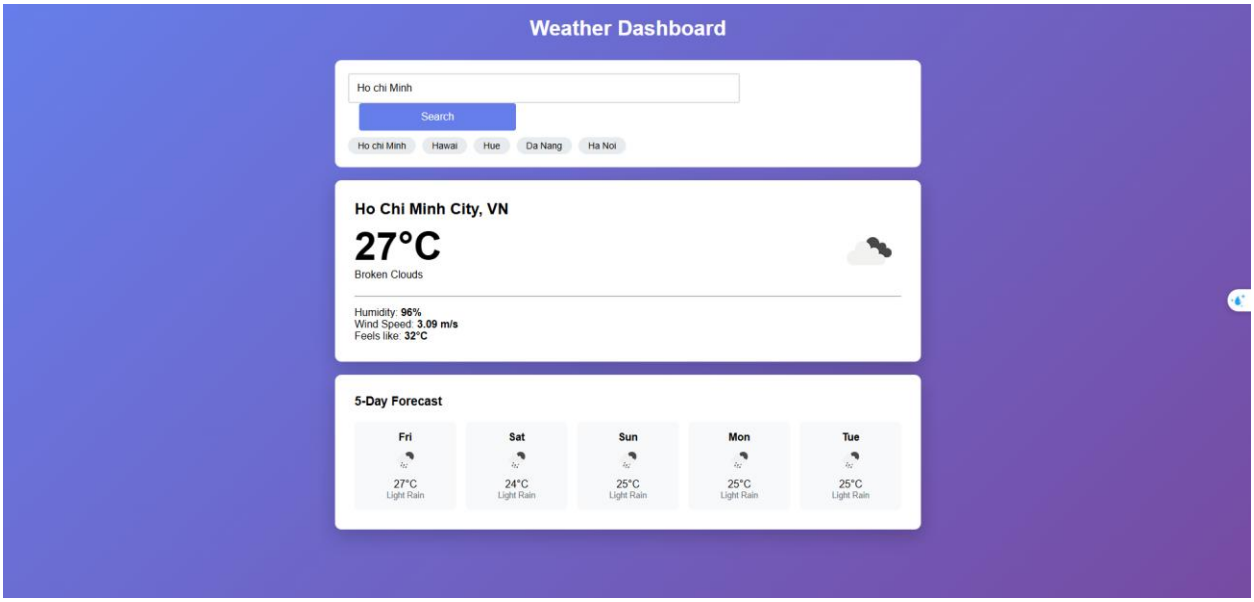
Course: Web Application Development

Lab 3: JavaScript Fundamentals

Github: https://github.com/TienAnh0108/Lab3_Assignment2.git

Exercise 2-1

Output:



1. API key
    a. What is the purpose of API key?
    - Stores the unique authentication key for OpenWeatherMap.
    b. Why do we need to use it?
    - The API requires this key to authorize requests and grant access to weather data.
2. Weather URL
    a. What is the purpose of Weather URL?
    - Base URL for fetching current weather data.
    b. Why do we need to use it?
    - Standardizes the endpoint for all current weather API calls.

3. Forecast URL
    a. What is the purpose of Forecast URL?
    - Base URL for fetching 5-day/3-hour forecast data.
    b. Why do we need to use it?
    - Standardizes the endpoint for all forecast API calls.
4. cityInput
    a. What is the purpose of cityInput value?
    - Reference to the city name input field.
    b. Why do we need to use it?
    - To read the user's search query (city name).
5. weatherDisplay
    a. What is the purpose of weatherDisplay value?
    - Reference to the area where current weather is shown.
    b. Why do we need to use it?
    - The target element for the displayWeather() function.
6. forecastDisplay
    a. What is the purpose of forecastDisplay value?
    - Reference to the area where the 5-day forecast grid is shown.
    b. Why do we need to use it?
    - The target element for the displayForecast() function's HTML output, ensuring the forecast appears separate from the current weather.
7. errorMessageDiv
    a. What is the purpose of errorMessageDiv value?
    - Reference to the dedicated area for displaying errors.
    b. Why do we need to use it?
    - To separate error messages from the main data display and apply specific error styling (red background, etc.).
8. recentSearchesDiv
    a. What is the purpose of recentSearchesDiv value?
    - Reference to the container for clickable recent search buttons.
    b. Why do we need to use it?
    - The target element for loadRecentSearches() to dynamically inject the recent city names, providing easy search recall.

```
const API_KEY = 'c3dab1f34c726169b85d6463a1d47e61'; // Get from openweathermap.org
const WEATHER_URL = 'https://api.openweathermap.org/data/2.5/weather';
const FORECAST_URL = 'https://api.openweathermap.org/data/2.5/forecast';

const cityInput = document.getElementById('cityInput');
const weatherDisplay = document.getElementById('weatherDisplay');
const forecastDisplay = document.getElementById('forecastDisplay');
const errorMessageDiv = document.getElementById('errorMessage');
const recentSearchesDiv = document.getElementById('recentSearches');
```

9. clearDisplay()
    a. What is the purpose of clearDisplay() function?
    - Clears displayed information and shows a loading state.
    b. Why do we need to use it?
    - To show a loading state and clear previous data/errors before a new search begins.
    c. How do we use it?
    - Sets the weatherDisplay HTML to a loading message (...).

```
function clearDisplay(message = 'Loading...') {
  weatherDisplay.innerHTML = `<div class="loading">${message}</div>`;
  forecastDisplay.innerHTML = '';
  clearError();
}
```

10. showError()
    a. What is the purpose of showError() function?
    - Displays an error message to the user.
    b. Why do we need to use it?
    - To inform the user of **API failures** (e.g., City not found) in a dedicated area.
    c. How do we use it?
    - Sets the error message text in errorMessageDiv with the .error CSS class

```
function showError(message) {
  errorMessageDiv.className = 'error';
  errorMessageDiv.innerHTML = message;
  weatherDisplay.innerHTML = '';
  forecastDisplay.innerHTML = '';
}
```

11. fetchWeather()
    a. What is the purpose of fetchWeather() function?
    - Asynchronously fetches the current weather data.
    b. Why do we need to use it?
    - To get the most up-to-date conditions (temperature, wind, humidity, etc.).
    c. How do we use it?
    - Uses the Fetch API and handles HTTP 404 (City not found) specifically.

```
async function fetchWeather(city) {
  const url = `${WEATHER_URL}?q=${encodeURIComponent(city)}&appid=${API_KEY}&units=metric&lang=en`;
  try {
    const response = await fetch(url);
    if (!response.ok) {
      if (response.status === 404) {
        throw new Error(`City not found: ${city}. Please check the name.`);
      }
      throw new Error(`HTTP Error: ${response.status}.`);
    }
    return await response.json();
  } catch (error) {
    throw error;
  }
}
```

12. fetchForecast()
    a. What is the purpose of fetchForecast() function?
    - Asynchronously fetches the 5-day/3-hour forecast data.
    b. Why do we need to use it?
    - To display future weather trends required for the forecast grid display.
    c. How do we use it?
    - Uses the Fetch API for the forecast endpoint.

```
async function fetchForecast(city) {
  const url = `${FORECAST_URL}?q=${encodeURIComponent(city)}&appid=${API_KEY}&units=metric&lang=en`;
  try {
    const response = await fetch(url);
    if (!response.ok) {
      throw new Error(`HTTP Error: ${response.status} while fetching forecast.`);
    }
    return await response.json();
  } catch (error) {
    throw error;
  }
}
```

13. displayWeather()
    a. What is the purpose of displayWeather() function?
    - Renders the current weather details onto the dashboard.
    b. Why do we need to use it?
    - To present the fetched data (temp, icon, description, etc.) to the user.
    c. How do we use it?
    - Constructs an HTML string to format all current weather metrics.

```
function displayWeather(data) {
  const tempC = Math.round(data.main.temp);
  const description = data.weather[0].description;
  const iconCode = data.weather[0].icon;
  const iconUrl = `https://openweathermap.org/img/wn/${iconCode}@2x.png`;

  weatherDisplay.innerHTML = `
    <div class="weather-card">
        <h2>${data.name}, ${data.sys.country}</h2>
        <div class="current-weather">
            <div>
                <span class="temp-display">${tempC}°C</span>
                <p style="text-transform: capitalize;">${description}</p>
            </div>
            <div>
                <img src="${iconUrl}" alt="${description}" width="100">
            </div>
        </div>
        <hr style="margin: 15px 0;">
        <p>Humidity: <strong>${data.main.humidity}%</strong></p>
        <p>Wind Speed: <strong>${data.wind.speed} m/s</strong></p>
        <p>Feels like: <strong>${Math.round(data.main.feels_like)}°C</strong></p>
    </div>
`;
}
```

14. displayForecast()
    a. What is the purpose of displayForecast() function?
    - Renders the simplified 5-day forecast grid.
    b. Why do we need to use it?
    - To filter the raw 3-hour data to show only one, relevant entry per future day.
    c. How do we use it?
    - Filters the raw list to select one data point per future day and maps it into HTML cards.

```
function displayForecast(data) {
  const forecastList = data.list;
  const dailyForecasts = {};
  const today = new Date().toLocaleDateString('en-US');

  for (let item of forecastList) {
    const date = new Date(item.dt * 1000);
    const dateStr = date.toLocaleDateString('en-US');

    if (dateStr === today) continue;

    if (!dailyForecasts[dateStr] && date.getHours() >= 10 && date.getHours() <= 14) {
      dailyForecasts[dateStr] = item;
    } else if (!dailyForecasts[dateStr]) {
      dailyForecasts[dateStr] = item;
    }
  }

  const forecastCards = Object.values(dailyForecasts).slice(0, 5).map(item => {
    const date = new Date(item.dt * 1000);
    const dayName = date.toLocaleDateString('en-US', { weekday: 'short' });
    const iconCode = item.weather[0].icon;
    const temp = Math.round(item.main.temp);
```

```
    return `
        <div class="forecast-item">
            <p><strong>${dayName}</strong></p>
            <img src="https://openweathermap.org/img/wn/${iconCode}.png" alt="Icon">
            <p>${temp}°C</p>
            <p style="font-size: 0.8em; color: #6c757d; text-transform: capitalize;">${item.weather[0].description}</p>
        </div>
    `;
  }).join('');

  forecastDisplay.innerHTML = `
    <div class="weather-card">
        <h3>5-Day Forecast</h3>
        <div class="forecast-grid">
            ${forecastCards}
        </div>
    </div>
`;
}
```

15. saveRecentSearch()
    a. What is the purpose of saveRecentSearch() function?
    - Adds a successfully searched city to a list of recent queries.
    b. Why do we need to use it?
    - To provide the user with quick access to their previous searches, improving usability.
    c. How do we use it?
    - Stores an array of cities in localStorage (max 5 cities), handles duplicates, and updates the display

```
function saveRecentSearch(city) {
  const cityClean = city.trim();
  let searches = JSON.parse(localStorage.getItem('recentCities') || '[]');

  searches = searches.filter(c => c.toLowerCase() !== cityClean.toLowerCase());

  searches.unshift(cityClean);

  if (searches.length > 5) {
    searches.pop();
  }

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

16. loadRecentSearches()
    a. What is the purpose of loadRecentSearches() function?
    - Retrieves and displays recent searches as clickable buttons.
    b. Why do we need to use it?

- To populate the recent search buttons when the page loads or after a new city is saved.

c. How do we use it?

- Reads the array from localStorage, creates clickable elements, and attaches an onclick handler.

```
function loadRecentSearches() {
  let searches = JSON.parse(localStorage.getItem('recentCities') || '[]');
  recentSearchesDiv.innerHTML = '';

  searches.forEach(city => {
    const cityElement = document.createElement('span');
    cityElement.className = 'recent-city';
    cityElement.textContent = city;
    cityElement.onclick = () => {
      cityInput.value = city;
      searchWeather();
    };
    recentSearchesDiv.appendChild(cityElement);
  });
}
```

17. searchWeather()

a. What is the purpose of searchWeather() function?

- The primary function that orchestrates the entire search process.

b. Why do we need to use it?

- To handle user input, manage the loading state, call the necessary fetch functions, and handle the final output or any errors.

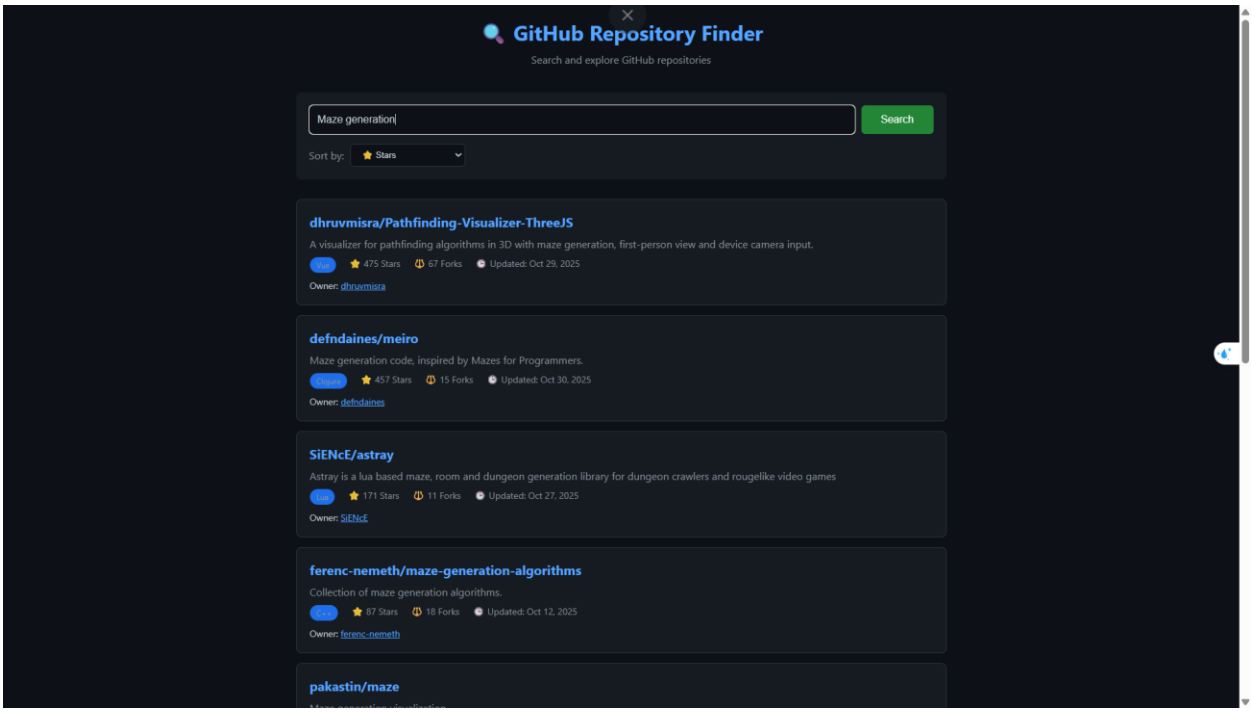
c. How do we use it?

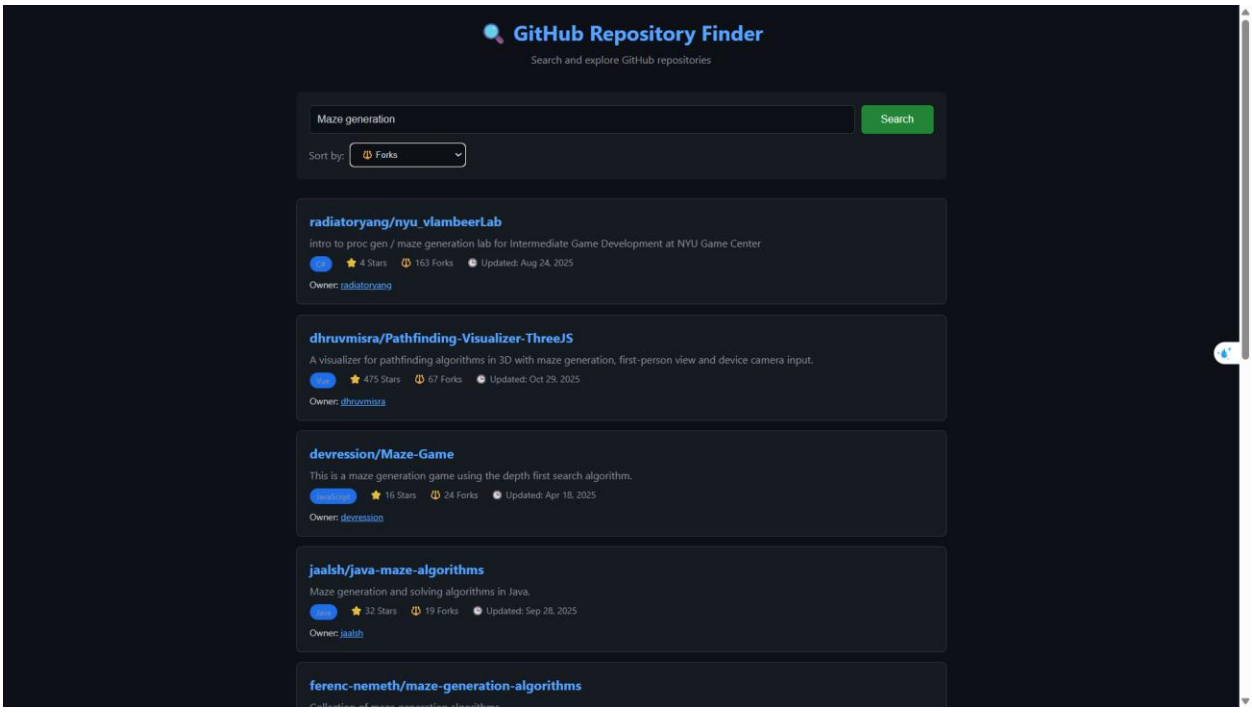
- Uses try...catch to manage asynchronous calls and routes errors to showError().

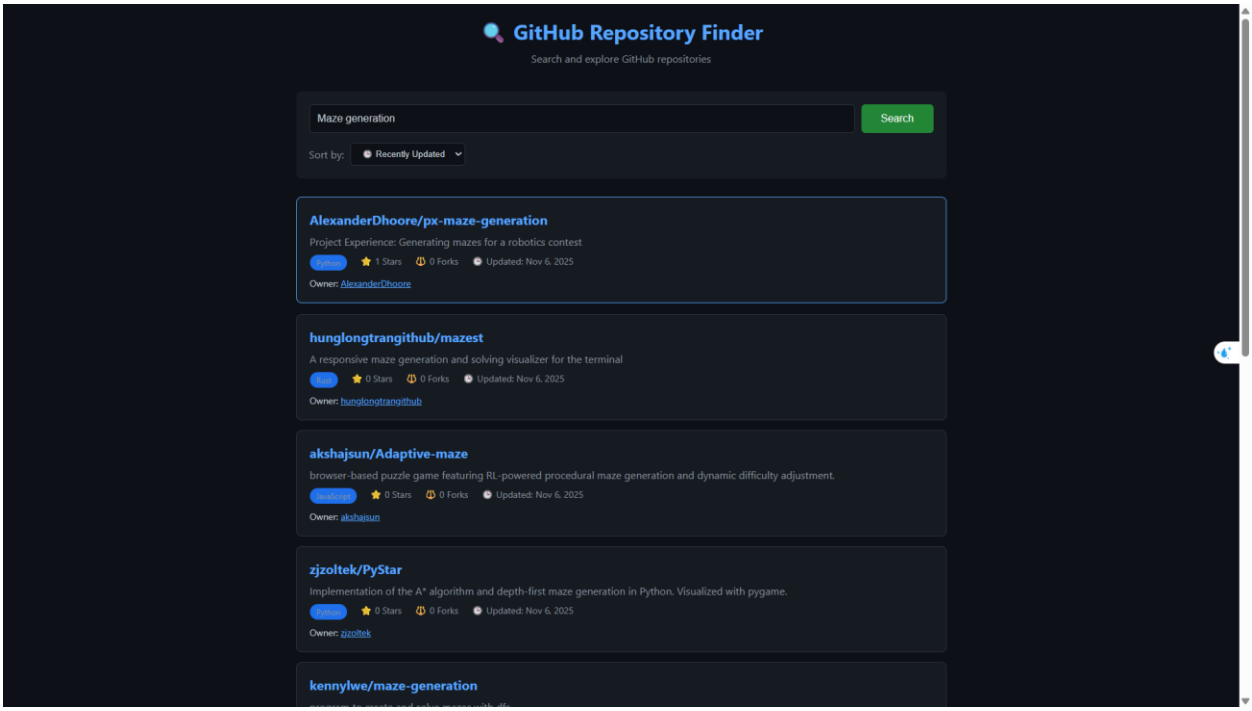
```
async function searchWeather() {
  const city = cityInput.value.trim();
  if (!city) {
    showError('Please enter a city name.');
    return;
  }

  clearDisplay();

  try {
    const weatherData = await fetchWeather(city);
    displayWeather(weatherData);

    const forecastData = await fetchForecast(city);
    displayForecast(forecastData);

    saveRecentSearch(city);

  } catch (error) {
    showError(error.message);
  }
}
loadRecentSearches();
```

Exercise 2-2

Output:







1. currentPage, currentQuery, totalResults
    a. What is the purpose of currentPage, currentQuery and totalResults value?
    - Global variables tracking the current page number, the active search keyword, and the total number of repositories found.
    b. Why do we need to use it?
    - To manage the pagination state across multiple function calls (searchRepositories, loadMore) and determine when to show the "Load More" button.
    c. How do we use it?

```
let currentPage = 1;
let currentQuery = '';
let totalResults = 0;
```

2. perPage, GITHUB_API_URL
    a. What is the purpose of perPage and GITHUB_API_URL?
    - Constants defining the number of results per page and the base URL for the GitHub Search API.
    b. Why do we need to use it?
    - To adhere to the API guidelines (per_page=10) and standardize the API endpoint for fetching repositories.
    c. How do we use it?

- Defined as global const variables.

```
const perPage = 10;
const GITHUB_API_URL = 'https://api.github.com/search/repositories';
```

3. searchInput, sortSelect
    a. What is the purpose of searchInput and sortSelect?
    - References to the search input field and the sort selection dropdown.
    b. Why do we need to use it?
    - To read the user's input (query) and the chosen sort criteria (stars, forks, or updated).
    c. How do we use it?
    - Uses document.getElementById() to link to the respective HTML elements.

```
const searchInput = document.getElementById('searchInput');
const sortSelect = document.getElementById('sortSelect');
```

4. repoList
    a. What is the purpose of repoList?
    - Reference to the container where the repository cards are displayed.
    b. Why do we need to use it?
    - The main target area for rendering search results using displayRepositories().
    c. How do we use it?
    - Uses document.getElementById('repoList').

```
const repoList = document.getElementById('repoList');
```

5. loadMoreContainer
    a. What is the purpose of loadMoreContainer?
    - Reference to the container holding the "Load More" button or loading message.
    b. Why do we need to use it?
    - To dynamically show, hide, or update the pagination control separate from the main result list.
    c. How do we use it?
    - Uses document.getElementById('loadMoreContainer').

```
const loadMoreContainer = document.getElementById('loadMoreContainer');
```

6. errorMessageDiv
    a. What is the purpose of errorMessageDiv?
    - Reference to the dedicated area for displaying error and alert messages.
    b. Why do we need to use it?
    - To inform the user of API failures or required input (e.g., "Please enter a search keyword.").
    c. How do we use it?
    - document.getElementById('errorMessage').

```
const errorMessageDiv = document.getElementById('errorMessage');
```

7. searchRepositories(query, sort, page)
    a. What is the purpose of searchRepositories(query, sort, page) function?
    - Asynchronously fetches a list of repositories from the GitHub API based on the search parameters.
    b. Why do we need to use it?
    - This is the primary function to interact with the external service and retrieve the raw data necessary for display.
    c. How do we use it?
    - Uses the Fetch API to call the GitHub URL. It includes a try...catch block to handle network errors and specifically checks for the Rate Limit (Status 403) error, calling showError() if an issue occurs

```
async function searchRepositories(query, sort = 'stars', page = 1) {
    const url = `${GITHUB_API_URL}?q=${encodeURIComponent(query)}&sort=${sort}&order=desc&page=${page}&per_page=${perPage}`;

    if (page === 1) {
        repoList.innerHTML = '<div class="loading">Searching...</div>';
    }
    clearError();

    try {
        const response = await fetch(url);
        // Handle Rate Limit (status 403)
        if (response.status === 403 && response.headers.get('X-RateLimit-Remaining') === '0') {
            throw new Error('GitHub API Rate Limit exceeded (60 req/hour). Please wait and try again.');
        }
        if (!response.ok) {
            throw new Error(`HTTP Error: ${response.status} - ${response.statusText}`);
        }
        const data = await response.json();
        return data;
    } catch (error) {
        console.error('API Error:', error);
        showError(error.message || 'An unknown error occurred while searching repositories.');
        repoList.innerHTML = '';
        loadMoreContainer.innerHTML = '';
        return null;
    }
}
```

8. displayRepositories(repos, append)
    a. What is the purpose of displayRepositories(repos, append) function?
    - Manages the rendering of search results.
    b. Why do we need to use it?
    - To either replace all previous results (new search) or add new results to the bottom (load more).
    c. How do we use it?
    - Check the append flag. If false, it clears repoList.innerHTML. It then calls appendRepositories() to perform the actual rendering.

```
function displayRepositories(repos, append = false) {
    if (!append) {
        repoList.innerHTML = '';
    }
    if (repos.length === 0 && currentPage === 1) {
        repoList.innerHTML = '<div class="loading">No repositories found for this query.</div>';
        loadMoreContainer.innerHTML = '';
        return;
    }
    appendRepositories(repos);
}
```

9. createRepoCard(repo) function
    a. What is the purpose of createRepoCard(repo) function?
    - Generates the complete HTML structure for a single repository item.
    b. Why do we need to use it?
    - To format the raw repository data into a visually structured and informative card, meeting the requirement to display name, description, stars, forks, language, owner, and link.
    c. How do we use it?
    - Constructs an HTML string (using template literals) containing data points like repo.full_name, repo.stargazers_count, and the owner.login.

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

    const updatedDate = new Date(repo.updated_at).toLocaleDateString('en-US', { year: 'numeric', month: 'short', day: 'numeric' });

    card.innerHTML = `
<a href="${repo.html_url}" target="_blank" class="repo-name">${repo.full_name}</a>
<p class="repo-description">${repo.description || 'No description provided.'}</p>

<div class="repo-meta">
    ${repo.language ? `<span class="language-badge">${repo.language}</span>` : ''}
    <span>⭐ ${formatNumber(repo.stargazers_count)} Stars</span>
    <span>🔱 ${formatNumber(repo.forks_count)} Forks</span>
    <span>🕒 Updated: ${updatedDate}</span>
</div>

<p style="margin-top: 10px; font-size: 14px;">
    Owner: <a href="${repo.owner.html_url}" target="_blank" style="color: #58a6ff;">${repo.owner.login}</a>
</p>
`;
    return card;
}
```

10. appendRepositories(repos) function
    a. What is the purpose of appendRepositories(repos) function?
    - Iterates through an array of repositories and adds them to the list.
    b. Why do we need to use it?
    - A utility function to decouple the logic of adding elements from the logic of managing the display state (displayRepositories).
    c. How do we use it?
    - Loops through the repos array and uses repoList.appendChild(createRepoCard(repo)) for each item.

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

11. performSearch() function
    a. What is the purpose of performSearch()?
    - Orchestrates a new search query or a search after a sort option change.
    b. Why do we need to use it?
    - To handle user interaction, validate the input, reset the pagination state (currentPage = 1), and initiate the data fetching process.
    c. How do we use it?
    - Reads input/sort values, resets state variables, calls searchRepositories(), and updates the total count and the load button status.

```
async function performSearch() {
    const query = searchInput.value.trim();
    const sort = getValue();

    if (query === '') {
        showError('Please enter a search keyword.');
        repoList.innerHTML = '';
        loadMoreContainer.innerHTML = '';
        return;
    }

    // Reset state for a new search
    currentQuery = query;
    currentPage = 1;
    totalResults = 0;

    const data = await searchRepositories(query, sort, currentPage);

    if (data) {
        totalResults = data.total_count;
        displayRepositories(data.items, false);
        updateLoadMoreButton();
    } else {
        repoList.innerHTML = '';
        loadMoreContainer.innerHTML = '';
    }
}
```

12. loadMore() function
    a. What is the purpose of loadMore() function?
    - Fetches the next page of search results.
    b. Why do we need to use it?
    - To implement the Pagination feature, allowing the user to view more results without cluttering the page initially.
    c. How do we use it?
    - Increments currentPage, calls updateLoadMoreButton(true), fetches the next page via searchRepositories(), and calls displayRepositories(..., true) to append results.

```
async function loadMore() {
    if (!currentQuery) return;

    currentPage++;
    const sort = getValue();

    updateLoadMoreButton(true); // Show loading state

    const data = await searchRepositories(currentQuery, sort, currentPage);

    if (data) {
        displayRepositories(data.items, true);
        updateLoadMoreButton();
    } else {
        currentPage--;
        updateLoadMoreButton();
    }
}
```

13. showError(message)
    a. What is the purpose of showError(message) function?
    - Displays an error message in the dedicated error area.
    b. Why do we need to use it?
    - To provide clear feedback to the user on validation or API failures.
    c. How do we use it?
    - Sets the errorMessageDiv.className to 'error' and inserts the message text.

```
function showError(message) {
    errorMessageDiv.className = 'error';
    errorMessageDiv.innerHTML = message;
}
```

14. clearError()
    a. What is the purpose of clearError() function?
    - Removes any active error messages.
    b. Why do we need to use it?
    - To clear the alert area when a new search begins or a previous error is resolved.
    c. How do we use it?
    - Sets errorMessageDiv.innerHTML to an empty string.

```
function clearError() {
    errorMessageDiv.innerHTML = '';
}
```

15. formatNumber(num)
    a. What is the purpose of formatNumber(num) function?
    - Formats large integers into a more readable format (e.g., 15000 -> 15k).
    b. Why do we need to use it?
    - To present statistics (Stars, Forks) concisely, enhancing the UI cleanliness and readability, as is common practice on platforms like GitHub.
    c. How do we use it?
    - Check if num is ≥ 1000. If true, calculates the value divided by 1000, formats it to one decimal place, and appends 'k'.

```
function formatNumber(num) {
    // TODO: Format large numbers (e.g., 1500 -> 1.5k)
    if (num >= 1000) {
        return (num / 1000).toFixed(1) + 'k';
    }
    return num;
}
function getValue() {
    return sortSelect.value;
}
```

16. getValue()
    a. What is the purpose of getValue() function?
    - Returns the currently selected sort option. (Corresponds to getSortValue()).
    b. Why do we need to use it?
    - To abstract the process of reading the dropdown value, making performSearch() and loadMore() cleaner.
    c. How do we use it?
    - Returns sortSelect.value.

```
function getValue() {
    return sortSelect.value;
}
```

17. updateLoadMoreButton(isLoading)
    a. What is the purpose of updateLoadMoreButton(isLoading) function?
    - Controls the visibility and text of the pagination control.
    b. Why do we need to use it?
    - To implement the **pagination logic** based on the hint: "Check total_count to show Load More button," and to handle the maximum 1000 results limit.
    c. How do we use it?
    - Compares loadedCount (currentPage * perPage) against totalResults and the hard limit (1000). Dynamically inserts the "Load More" button or a loading message.