Lab 8: AJAX

Task 1: Basic Text Loading

Simple Content Loading

1. Create an HTML page with a button and a display div
2. Load content from a local text file when the button is clicked
3. Show loading state while content is being fetched

Sample Implementation Steps:

1. Create data.txt with some sample text content
2. Create button and display div in HTML
3. Implement AJAX request using one of the methods:
    - XMLHttpRequest
    - jQuery $.load()
4. Add loading indicator

Task 2: Pokémon Search Card

Using the PokéAPI

Create a Pokémon search application that displays Pokémon information in a card format using the free PokéAPI

API Details

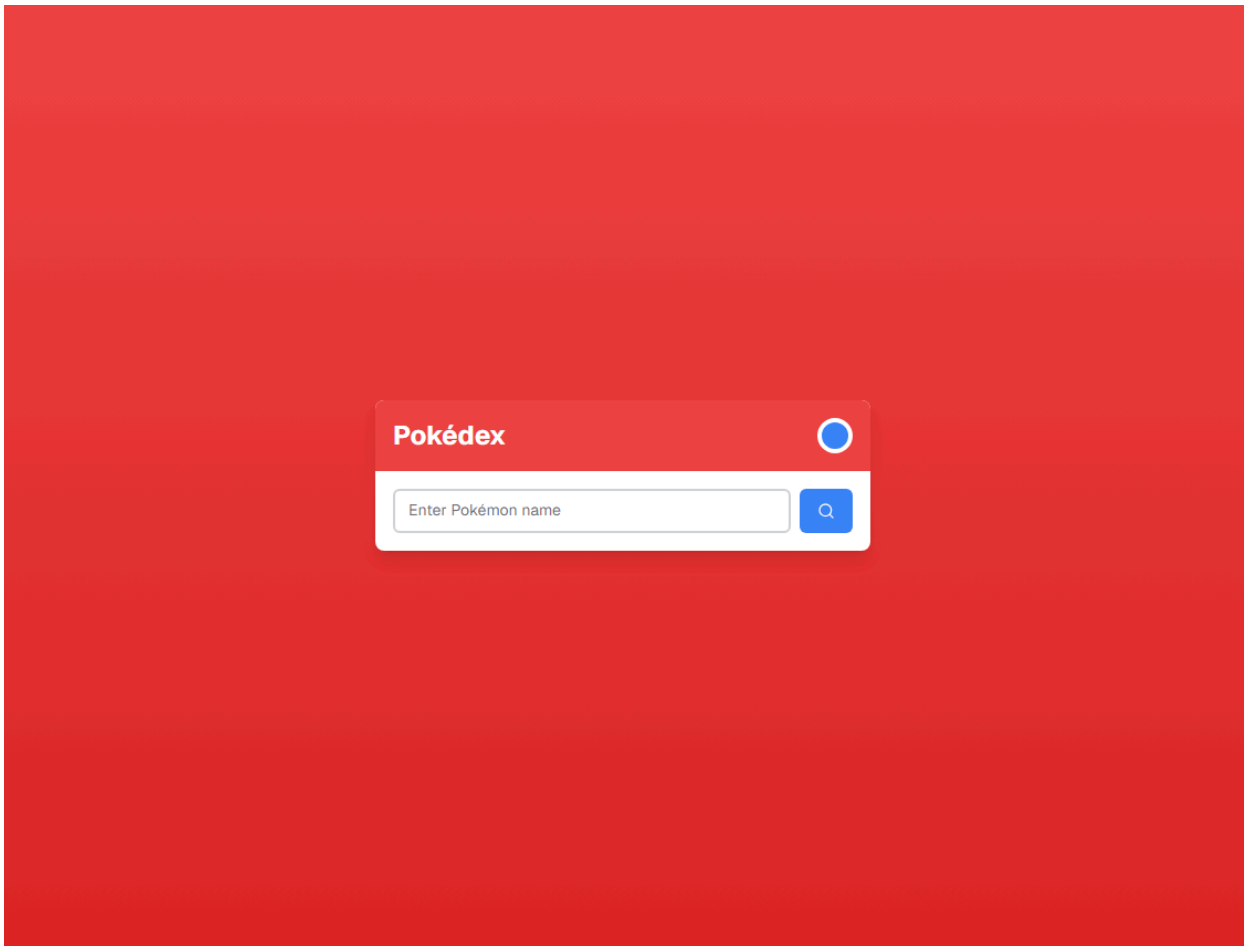
- Endpoint: `https://pokeapi.co/api/v2/pokemon/[name]`
- Example: `https://pokeapi.co/api/v2/pokemon/pikachu`

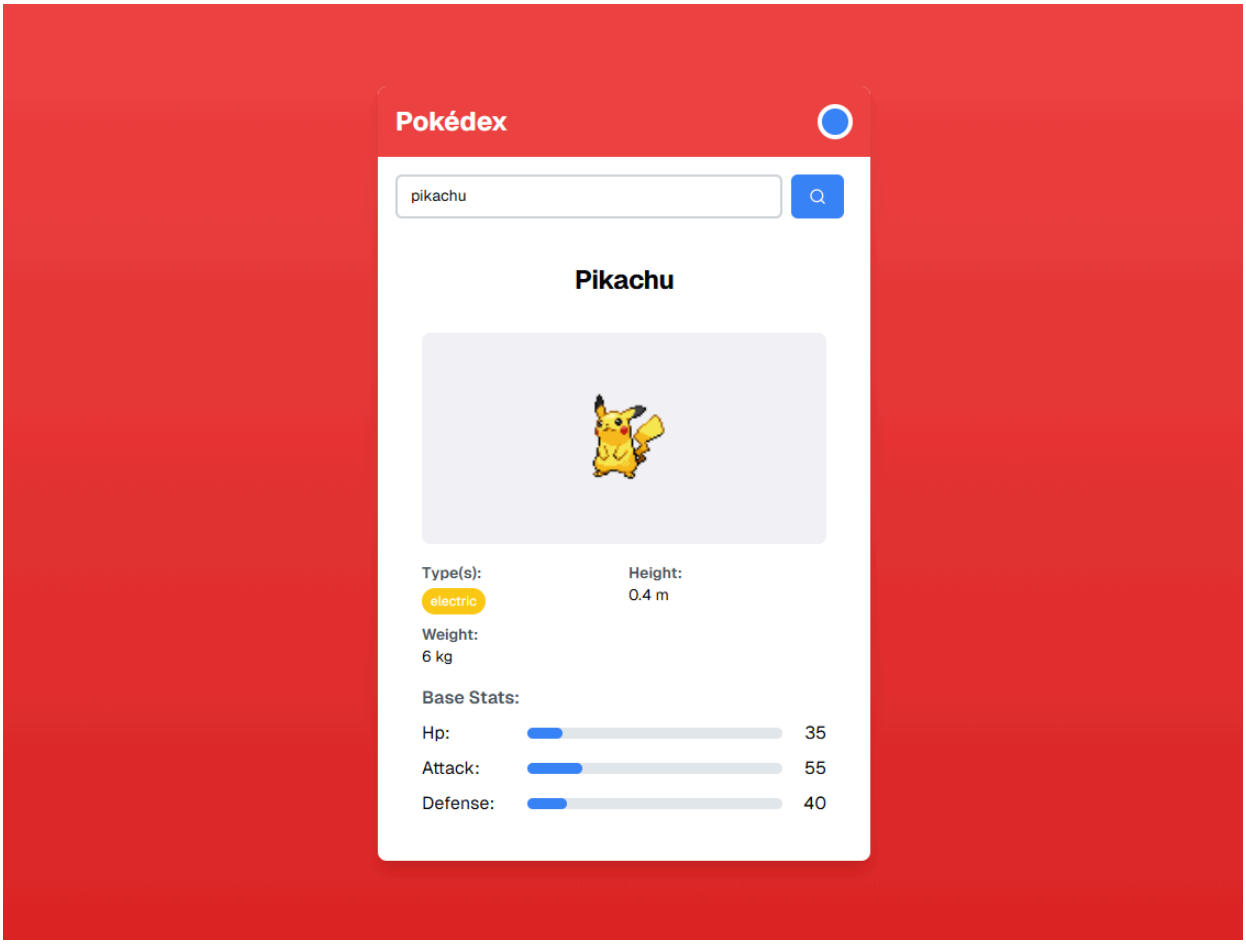
Requirements

1. Create a search input for Pokémon names
2. Display a card with:
    - Pokémon sprite/image
    - Name
    - Type(s)
    - Height and Weight

    - At least 3 base stats
3. Show loading state while fetching data
4. Handle errors for invalid Pokémon names
5. Make it mobile responsive

Screenshots





Task 3: User Directory

Working with JSON Data

Use the JSONPlaceholder API to create a user directory.

API Endpoint: `https://jsonplaceholder.typicode.com/users`

Main Layout

- Use a responsive **grid layout** for user cards. If the screen width is narrow (below 768px), switch to a list layout.
- Each user should be presented as a **card** with the following details:
    - **Name:** Displayed at the top as a header.
    - **Email:** Displayed under the name.
    - **Company Name:** Displayed below the email.
- The user cards should be aligned centrally with **equal spacing** between them

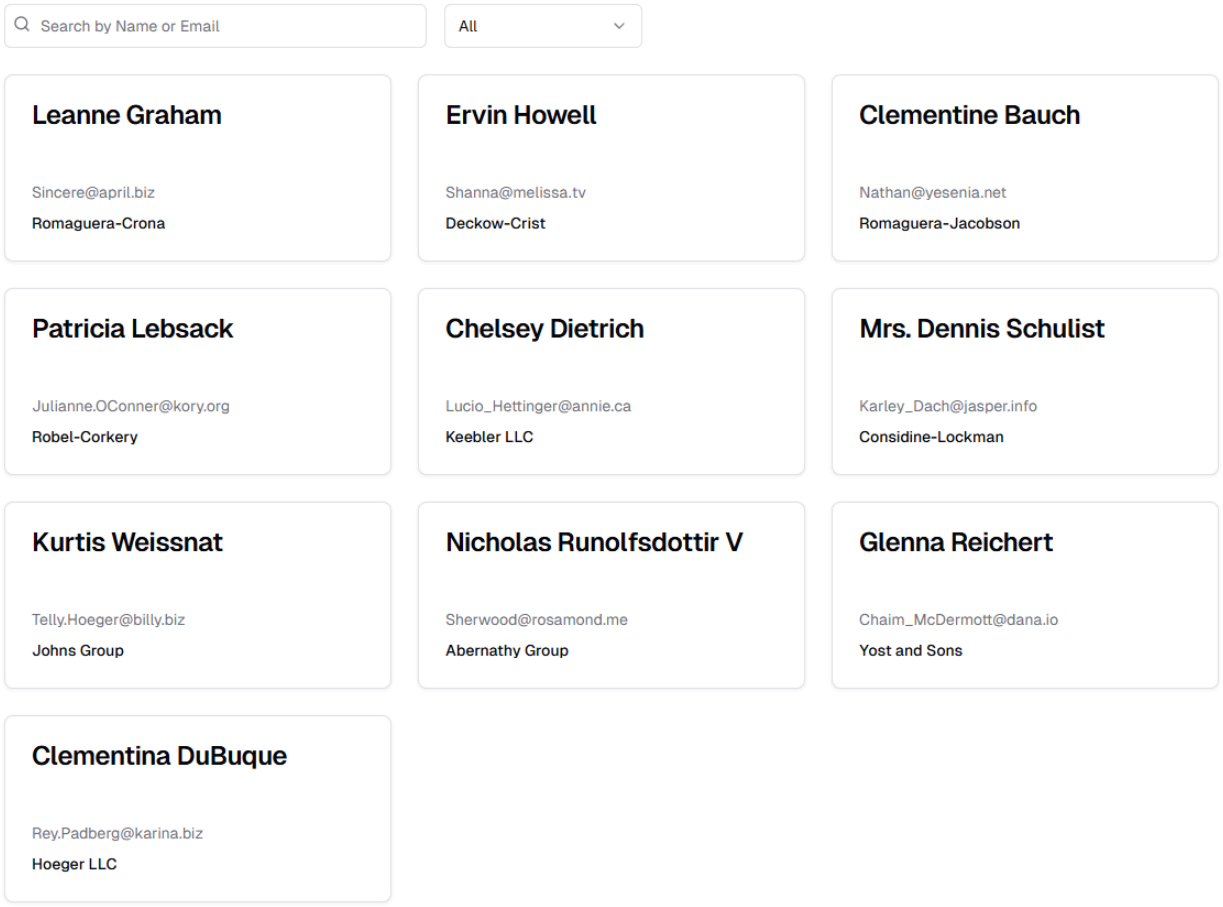
Search and Filter Functionality

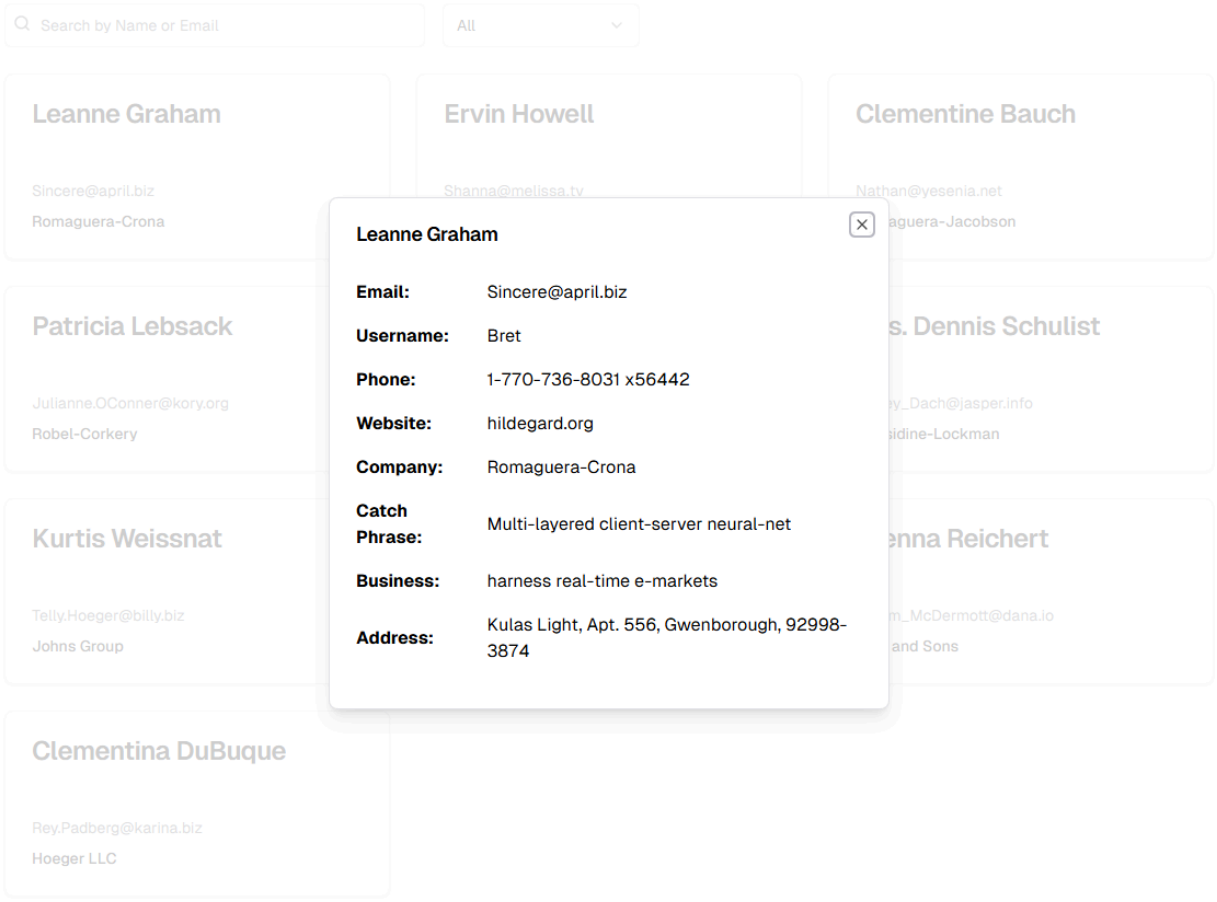
1. **Search Bar:**
    - Place a **centered search bar** at the top of the page.
    - Add a placeholder text: `"Search by Name or Email"`.
    - **Important:** Search functionality must **operate on the JavaScript array** containing the user data, not by querying or manipulating the DOM elements directly.
    - Implement a **case-insensitive search** for better user experience.
2. **Dropdown Filter:**
    - Add a **dropdown filter** next to the search bar to filter users by their **company**.
    - The dropdown should populate with **unique company names** fetched from the user data.
    - **Important:** Filtering should be done by first filtering the user array (not DOM manipulation) and then rendering the filtered results in the UI.
3. **Combined Search and Filter:**
    - Ensure that **both search and filter** work in combination:
        - For example, if a user searches for "John" and selects "TechCorp" from the dropdown, the displayed results should reflect users matching **both** criteria.
    - Re-render the entire user list from the filtered array when a search or filter action occurs.

Detailed View

- When clicking on a user card, the page should show a detailed view on the same page or in a modal.
    - In the detailed view, the following information must be shown:
        - Full Name
        - Email
        - Username
        - Phone Number
        - Website
        - Company Name
        - Company Catch Phrase
        - Company Business
        - Address (in this format: Street, Suite, City, Zip-Code)
- Include a 'Back' button to return to the grid or list view.

Screenshots





Task 4: Post Gallery

Loading Images with AJAX

Use the JSONPlaceholder API for posts and images.

API Endpoint: `https://jsonplaceholder.typicode.com/photos`

Main Layout

- The gallery should display images in a **responsive grid layout**:
  - For wide screens (above 1024px), display 4 images per row.
  - For medium screens (between 768px and 1024px), display 3 images per row.
  - For smaller screens (below 768px), switch to 2 images per row.
- Each image should be presented as a **card** with the following details:
  - **Image Thumbnail** (150px by 150px).
  - **Title** of the image below the thumbnail.

Load More Button

- Place the **"Load More" button** centered below the grid of images.
- When clicked, it should load additional images and append them to the current list.
- After all images are loaded, the button should disappear or be disabled with a message "No more images to load."

Loading Animation:

- While images are being loaded, show a **spinner animation** in place of the "Load More" button.
- The animation should be visible for a minimum of 1 second even if the data loads quickly, ensuring a smooth user experience.

Screenshots