Step 1: Setting Up the Basic Structure

- **Goal:** Create the HTML structure for the Address Book.
- **Tasks:**
  - Add a form with input fields for name, email, and phone, along with a submit button.
  - Add a search input field for filtering contacts.
  - Create a container (`div`) to display the contact list.
- **Practice:**
  - Write basic HTML with IDs: `contactForm`, `contactName`, `contactEmail`, `contactPhone`, `contactSubmitButton`, `contactList`, and `searchInput`.

Step 2: Initialize the Application

- **Goal:** Set up the `addressBook` object and its initial properties.
- **Tasks:**
  - Define the `contacts` array to store contact details.
  - Add flags `isEditing` and `editingEmail` to handle editing states.
  - Implement a `renderContacts` method to display an empty message initially.
- **Practice:**
  - Test rendering the empty state.

Step 3: Adding a Contact

- **Goal:** Implement functionality to add a contact.
- **Tasks:**
  - Write the `addContact` method to add a new contact.
  - Ensure contacts are stored in the `contacts` array.
  - Re-render the contact list after adding a contact.
- **Practice:**
  - Test adding multiple contacts and check the list updates dynamically.

Step 4: Displaying Contacts

- **Goal:** Render the contact list dynamically in the DOM.
- **Tasks:**
  - Update the `renderContacts` method to display name, email, and phone for each contact.

    - Include buttons for editing and deleting contacts.
- **Practice:**
    - Style the contact list for better appearance using CSS.

Step 5: Editing a Contact

- **Goal:** Implement functionality to edit contact details.
- **Tasks:**
    - Write the `editContact` method to populate the form with the contact's current details.
    - Disable the email input to prevent changes to the email address.
    - Update the form's submit button text to "Update Contact."
- **Practice:**
    - Test editing different contacts and ensure the form updates accordingly.

Step 6: Updating a Contact

- **Goal:** Update an existing contact's details.
- **Tasks:**
    - Write the `updateContact` method to modify the contact's name and phone in the `contacts` array.
    - Re-render the contact list after the update.
    - Reset the form after updating the contact.
- **Practice:**
    - Test updating various contacts and verify the changes reflect correctly.

Step 7: Deleting a Contact

- **Goal:** Implement functionality to delete a contact.
- **Tasks:**
    - Write the `deleteContact` method to remove the contact from the `contacts` array.
    - Re-render the contact list after deletion.
- **Practice:**
    - Test deleting contacts and ensure the list updates dynamically.

Step 8: Searching Contacts

- **Goal:** Allow users to search for contacts by name or email.
- **Tasks:**
  - Write the `searchContacts` method to filter contacts based on the query.
  - Update the displayed contacts dynamically as the user types in the search input.
- **Practice:**
  - Test searching for contacts using partial and full matches.

Step 9: Sorting Contacts

- **Goal:** Sort the contact list alphabetically by name.
- **Tasks:**
  - Write the `sortContacts` method to sort the `contacts` array.
  - Re-render the contact list after sorting.
- **Practice:**
  - Test sorting contacts after adding multiple entries.

Step 10: Resetting the Form

- **Goal:** Clear the form after adding or updating a contact.
- **Tasks:**
  - Write the `resetForm` method to reset all input fields and editing states.
  - Re-enable the email input field for new contacts.
- **Practice:**
  - Test adding and editing contacts to ensure the form resets properly.

Step 11: Polishing the UI

- **Goal:** Improve the user interface.
- **Tasks:**
  - Style the contact list, buttons, and form for a professional appearance.
  - Add hover effects for buttons.
  - Use responsive design techniques to ensure usability on mobile and desktop devices.
- **Practice:**
  - Test the application on different devices and screen sizes.

Bonus Steps

1. **Add Local Storage:**
    - Save contacts to `localStorage` to persist data between sessions.
    - Load contacts from `localStorage` on page load.
2. **Export Contacts:**
    - Add functionality to export the contact list as a CSV or JSON file.
3. **Dark Mode:**
    - Add a toggle for light and dark themes.
4. **Pagination:**
    - If the contact list grows, implement pagination to display contacts in chunks.

Practicing Each Step Focus on implementing and testing each step independently. After completing all steps, integrate the functionality into a fully functional Address Book Application.