Prerequisites for Developing the Contacts Management System

Here's a breakdown of the prerequisites and considerations for effectively developing the provided contacts management system.

---

1. Technical Skills

JavaScript Concepts

1. **Array Manipulation**:
    - Use of `map`, `find`, `filter` to manage and update the `contacts` array.
    - Handling updates and deletions dynamically.
2. **Regular Expressions**:
    - Search functionality using `RegExp` for case-insensitive matching of names and emails.
3. **DOM Manipulation**:
    - Dynamic rendering of contacts with `innerHTML` and `createElement`.
    - Attaching event listeners to dynamically created elements.
4. **Event Handling**:
    - Capturing events for adding, editing, deleting, and searching contacts.
5. **Web Workers**:
    - Using workers for bulk updates without blocking the main thread.
6. **Async Programming**:
    - Fetching contacts data using `fetch` and `async/await`.
    - Graceful error handling for API calls.

---

2. Functional Requirements

Core Features

1. **Fetch Contacts**:
    - Retrieve contacts from the JSONPlaceholder API and populate the UI.

2. **Add Contact**:
    - Allow users to manually add new contacts with `name` and `email`.
3. **Edit Contact**:
    - Enable inline editing of contact details (name and email).
4. **Delete Contact**:
    - Remove a contact from the list dynamically.
5. **Search Contacts**:
    - Filter contacts by name or email in real-time.
6. **Bulk Updates**:
    - Use a Web Worker to perform bulk updates efficiently.

---

3. UI Requirements

Input Fields

1. **Add Contact**:
    - Text inputs for `Name` and `Email`.
    - A button to add the contact.
2. **Search**:
    - A search bar for filtering contacts.

Contacts List

- Display each contact with:
    - `Name` and `Email`.
    - Buttons for **Edit** and **Delete** actions.

Bulk Update Button

- A button to trigger the Web Worker for bulk updates.

Styling

- Use basic CSS or frameworks like **Tailwind CSS** for:
    - Layout (`flex`, `justify-between`).
    - Buttons (`bg-yellow-500`, `bg-red-500`).

---

4. Key Functions

1. Fetch Contacts

- Fetch data from the JSONPlaceholder API.
- Map API response to a consistent structure (`id`, `name`, `email`).

2. Render Contacts

- Render the list of contacts dynamically.
- Reflect changes (add, edit, delete) immediately.

3. Add Contact

- Validate inputs for `name` and `email`.
- Add the new contact to the `contacts` array and update the UI.

4. Edit Contact

- Prompt the user to update contact details.
- Validate and apply changes to the `contacts` array.

5. Delete Contact

- Remove the contact from the `contacts` array.
- Re-render the updated list.

6. Search Contacts

- Filter contacts dynamically using a case-insensitive search term.

7. Bulk Updates with Web Workers

- Perform email updates in bulk using a worker for non-blocking updates.
- Post the updated contacts back to the main thread and re-render them.

---

5. Development Workflow

1. **Setup Data and Fetch Contacts**:
    - Define the `contacts` array.
    - Fetch and initialize contacts from the API on page load.

2. **Dynamic Rendering**:
    - Implement `renderContacts` to handle rendering updates efficiently.
3. **Add/Edit/Delete Operations**:
    - Allow users to add, edit, and delete contacts interactively.
4. **Search Functionality**:
    - Implement search with regular expressions for real-time filtering.
5. **Bulk Updates**:
    - Use Web Workers to perform bulk updates for email domains.
6. **Error Handling**:
    - Ensure robust error handling for invalid inputs and failed API requests.

---

6. Testing Scenarios

Functional Tests

1. **Fetch and Render Contacts**:
    - Verify contacts are fetched and displayed correctly.
2. **Add Contact**:
    - Test adding contacts with valid and invalid inputs.
3. **Edit Contact**:
    - Confirm changes are applied dynamically.
4. **Delete Contact**:
    - Ensure contacts are removed from the list and UI.
5. **Search**:
    - Test filtering contacts by name and email.
6. **Bulk Updates**:
    - Verify emails are updated efficiently using Web Workers.

Edge Cases

1. Adding a contact with missing or invalid details.
2. Editing a non-existent contact (if applicable).
3. Searching for contacts with no matches.

---

7. Optional Enhancements

Persistent Storage

- Save contacts to `localStorage` for persistence across page reloads.

Validation

- Add stricter validation for email formatting and duplicate entries.

Pagination

- Implement pagination for handling a large number of contacts.

UI Enhancements

- Add success or error notifications for actions like adding, editing, or deleting.

By meeting these prerequisites and following the outlined workflow, the contacts management system can be built and maintained effectively. Let me know if you'd like further guidance!