Angular Assessment: Contact Management with Local Storage

Objective:

Create an Angular application that allows users to manage a list of contacts using local storage for data persistence. The app should support Create, Read, Update, and Delete (CRUD) operations on contacts.

Contact Object Structure:

```
interface Contact {
  id: number;
  firstName: string;
  lastName: string;
  gender: string;  // Can be 'Male', 'Female', 'Other'
  dateOfBirth: string;  // ISO format date string 'YYYY-MM-DD'
  location: string;
  email: string;
  mobile: string;
}
```

Requirements:

1. **Project Setup:**
    - Create a new Angular 17 project.
    - Set up local storage to persist contact data.
2. **Data Service:**
    - Create a ContactService to handle all CRUD operations with local storage:
        - **Create Contact:** Add a new contact to local storage.
        - **Read Contacts:** Retrieve all contacts from local storage.
        - **Update Contact:** Update an existing contact in local storage.
        - **Delete Contact:** Remove a contact from local storage.
3. **Components:**
    - **Contact List Component:**
        - Displays a list of all contacts.

        - Each contact should show basic details like firstName, lastName, email, and mobile.
        - Include buttons for editing and deleting contacts.
    - **Contact Details Component:**
        - Displays full details of a selected contact.
        - Option to go back to the contact list.
    - **Contact Add/Edit Component:**
        - Display contact details for adding a new contact or editing an existing one.
        - Fields for each property in the contact object should be displayed directly using , , or other appropriate HTML elements for each data field.
        - Provide buttons to add a new contact or save changes to an existing contact.
        - A button to clear the input fields should be included.
4. **Routing:**
    - Use Angular Router to navigate between the contact list, contact details, and add/edit contact views.
5. **Local Storage Management:**
    - Implement methods in ContactService to handle saving, retrieving, updating, and deleting data in local storage.
    - Ensure that data is retained even after the page is refreshed.
6. **Error Handling:**
    - Implement basic error handling in case of invalid data inputs or operations (like trying to delete a non-existent contact).
7. **Styling:**
    - Use basic CSS to style the application. Emphasize a clean, simple UI.

Instructions:

1. **Initialization:**
    - Clone the repository and run npm install to install dependencies.
    - Use ng serve to start the development server.
2. **Tasks:**
    - Complete the ContactService to handle all CRUD operations.
    - Implement the components as described above.
    - Test the application by adding, viewing, editing, and deleting contacts.
3. **Submission:**

- Submit the complete Angular project, including all source code files and a README file explaining how to run the application.