**Process, Design, and Data Flow Guide**

**Business Overview**

**Why?**  
This application is a lightweight frontend-only simulation of a document and contact management system. It replicates a typical admin panel with login, dashboard (table), and contact management functionality, without relying on a backend server.

**Who?**

* **Users**: End users who will log in, view file data, and manage contacts.
* **Admin (optional)**: A role that may be implemented to differentiate access levels.
* **Developers/Testers**: Implement and verify the application based on requirements.

**Technology**

* Angular (with Reactive Forms)
* RxJS (for authentication and state management)
* Local Storage & Session Storage (for persistent data and session handling)
* JSON (used for storing initial dump of files and contacts)
* Firebase (optional: authentication or storage)
* HTML/CSS/Bootstrap (UI and styling)

**Process Flow**

**Module 1: Authentication (Login/Logout)**

**Brief:**  
Handles login/logout functionality and maintains session state via sessionStorage. Prevents unauthorized access to internal pages.

**Features**

1. **Login Form**
   * Validates credentials against local JSON file or session-stored data.
   * Stores login session on successful login.
   * Redirects to home/dashboard.
   * **Acceptance Criteria:**
     + Valid credentials allow access.
     + Invalid credentials show error message.
   * **Test Cases:**
     + ✅ Login with valid credentials
     + ❌ Login with invalid credentials
     + 🔒 Access home page without login
     + 🔓 Logout clears session and redirects
   * **Technical Details:**
     + RxJS BehaviorSubject for session state.
     + Route Guards to block unauthorized access.
   * **Compliances:**
     + Sensitive info should not be exposed in local/session storage.

**Module 2: Home Page (Files Table)**

**Brief:**  
Displays a table of files with filtering, pagination, and download/view actions.

**Features**

1. **Data Table**
   * File list loaded from a JSON file stored locally.
   * Search/Filter by file name/type.
   * Pagination controls (e.g., 10 items/page).
   * View (open file in new tab if viewable).
   * Download (direct link).
   * **Acceptance Criteria:**
     + File table loads correctly.
     + Filters and pagination work.
   * **Test Cases:**
     + Filter by type
     + Paginate file list
     + Click view -> opens file
     + Click download -> triggers download
   * **Technical Details:**
     + JSON loaded once, stored in memory.
     + Table component handles filter/paginate via pure frontend logic.

**Module 3: Contact Page (Add/Edit Form)**

**Brief:**  
Reactive form to collect contact information, supporting multiple emails, phones, and addresses.

**Features**

1. **Reactive Form**
   * Add/Edit contact.
   * Dynamic FormArray for multiple values.
   * Save to local storage.
   * **Acceptance Criteria:**
     + Add contact with multiple entries
     + Form validations trigger correctly
   * **Test Cases:**
     + Add multiple emails/phones/addresses
     + Invalid email shows error
     + Save stores data locally
   * **Technical Details:**
     + Angular Reactive Forms with FormArray
     + On submit: push data to localStorage
     + JSON structure for contacts stored locally

**Module 4: Contact Info Page (View/Edit Contacts)**

**Brief:**  
Displays a list of saved contacts from local storage. Allows editing each contact.

**Features**

1. **Editable Table**
   * Load contacts from local storage.
   * Edit button opens contact in Contact Form.
   * Updated data persists in local storage.
   * **Acceptance Criteria:**
     + Table loads contacts
     + Edit pre-fills form
     + Save updates correctly
   * **Test Cases:**
     + Edit existing contact
     + Save updated info
     + Add then immediately edit
   * **Technical Details:**
     + Contact data stored in localStorage['contacts']
     + Uses route parameters or shared RxJS Subject to send selected contact to edit form

**Third-Party Integration**

* No third-party backend. Firebase *may* be used optionally for auth/storage (define later if implemented).
* File links simulate actual downloads using dummy .pdf, .txt, .jpg in assets folder.

**Security Matrix**

| **Feature** | **Security Check** |
| --- | --- |
| Login | Credential check using sessionStorage |
| Route Guards | Prevents access without valid session |
| Contact Info | Only accessible when logged in |

*(Add Excel for detailed role-access if needed)*

**Notifications Matrix**

| **Trigger** | **Recipient** | **Method** |
| --- | --- | --- |
| On Save Contact | User | Toast popup |
| On Login Fail | User | Toast popup |
| On Download/View File | User | None (direct) |

*(Maintain Excel separately if required)*

**Engines (Optional)**

None required for now. RxJS-based authentication can be loosely considered a state engine.

**Data Migration**

* Not applicable – all data exists in JSON or local storage from the beginning.

**Data Design**

**Database Structure (local storage keys)**

json

CopyEdit

localStorage['contacts'] = [

{

"id": "UUID",

"name": "John Doe",

"emails": ["john@example.com", "j.doe@domain.com"],

"phones": ["1234567890", "0987654321"],

"date\_of\_birth": “-8-7-2002”

“gender”:”male”,

“password”:encrypted

},

...

];

localStorage['files'] = [

{

"id": "file1",

"name": "Document A",

"type": "pdf",

"size": "2MB",

"url": "/assets/files/doc-a.pdf",

“user\_id”:2,

“created\_on”:”20-05-2012”

},

...

];

**Table Descriptions**

* contacts: Stores user contact information.
* files: List of downloadable/viewable files.

**ER Diagram**

*(You can use tools like dbdiagram.io to represent the contact-file relationships — though no direct relationship exists.)*

**Data Dictionary**

| **Field** | **Description** | **Type** | **Format** | **Remarks** |
| --- | --- | --- | --- | --- |
| id | Unique Identifier | string | UUID | Autogenerated |
| name | Contact Name | string | Text | Required |
| emails | Email list | string[] | Email | At least 1 required |
| phones | Phone number list | string[] | Digits | Optional |
| addresses | Addresses list | string[] | Text | Optional |

**Project Estimation**

**Estimated Duration**

| **Phase** | **Timeframe** |
| --- | --- |
| Planning | 1 day |
| Development | 3 days |

**Glossary**

| **Term** | **Meaning** |
| --- | --- |
| Reactive Form | Form structure controlled via component logic |
| FormArray | Angular array of form controls for multiple entries |
| RxJS Subject | Event emitter used for state change communication |
| sessionStorage | Temp storage cleared on browser close |
| localStorage | Persistent client-side data storage |