# Indaba Care: Visual Guide for Non-Technical Team Members

<div align="center">
<img src="https://via.placeholder.com/500x150/4A90E2/FFFFFF?text=Indaba+Care" alt="Indaba Care Logo" width="500"/>
<p><em>Connecting families and caregivers, even when offline</em></p>
</div>

## What Are We Building?

Indaba Care is an app that helps parents and nannies manage childcare activities. Imagine it as a digital notebook where caregivers can:

- Record daily activities (meals, naps, learning)
- Take and share photos securely with family members
- Track schedules and milestones
- Access childcare resources
- Work even when the internet is down!

<div align="center">
<img src="https://via.placeholder.com/800x400/F5F5F5/333333?text=App+Screenshots" alt="App Screenshots" width="800"/>
<p><em>Example screens from the Indaba Care application</em></p>
</div>

## The Special Power: "Offline-First"

Most apps stop working when you lose internet connection. Indaba Care is different!

<div align="center">
<img src="https://via.placeholder.com/800x400/E8F5E9/333333" alt="Offline-First Diagram" width="800"/>

```
┌─────────────────────────┐     ┌─────────────────────────┐
┌─────────────────────────┐
│        │     │       │     │        │
│ **Local** Device   │ ◄──────────► │ Sync Engine  │ ◄──────────► │ Cloud **Server**  │
│ (PouchDB)     │     │       │     │ (Firebase)   │
│        │     │       │     │        │
└─────────────────────────┘     └─────────────────────────┘
└─────────────────────────┘

  Works          Manages          Available
 **ALWAYS**!        Synchronization       **when** online
```

<p><em>How offline-first architecture works in Indaba Care</em></p>
</div>

1. Everything you do gets saved to your device FIRST
2. When internet connection returns, it syncs with the cloud
3. This means parents and nannies in rural areas with spotty internet can still use the app!

Think of it like writing notes in a paper notebook (works offline), but those notes automatically copy themselves to a shared family album when internet becomes available.

## The Building Blocks

<div align="center">
<img src="https://via.placeholder.com/800x500/FFF8E1/333333" alt="Architecture Diagram" width="800"/>

```
┌──────────────────────────────────────────────────────────────────────────
──────────────────────────────────────┐
│                    INDABA CARE APP                          │
├──────────────────────────────────┬──────────────────────────────────────────────────────────┬─────
──────────────────────────────────────┤
│   FRONTEND      │        DATA HANDLING        │  BACKEND          │
│   (What users   │      (Business Logic)       │  (Cloud Services)   │
│     see)        │                             │                │
├──────────────────────────────────┼──────────────────────────────────────────────────────────┼─────
──────────────────────────────────────┤
│               │                          │                  │
│   React       │        PouchDB           │    Firebase        │
│   Next.js     │      (Local Database)       │   ┌──────────────────────────────┐      │
│   Chakra UI     │                          │  │ Authentication │      │
│               │           │               │   │               │     │
│               │           ▼               │   │   Firestore     │     │
│               │                          │   │               │     │
│               │       Sync Engine          │   │   Storage      │     │
│               │                          │   │               │     │
│               │           │               │   └──────────────────────────────┘      │
│               │           ▼               │                  │
└──────────────────────────────────┴──────────────────────────────────────────────────────────┴─────
──────────────────────────────────────┘
```

<p><em>Indaba Care Architecture Overview</em></p>
</div>

### 1. Firebase: Our Digital Filing Cabinet

**Firebase** is like a digital filing cabinet with several drawers:

<div align="center">
<img src="https://via.placeholder.com/800x300/E3F2FD/333333" alt="Firebase Components" width="800"/>

```
┌─────────────────────────────────────────────────────────────────────
─────────────────────┐
|                 FIREBASE                    |
├──────────────────────────────┬──────────────────────────────┬──────
─────────────────────┤
|   Authentication   |     Firestore     |     Storage      |
|   (Who are you?)    |  (What data exists?) |  (Where are files?) |
├──────────────────────────────┼──────────────────────────────┼──────
─────────────────────┤
|               |               |              |
| - User accounts     | - Users collection  | - Photos          |
| - Login/logout      | - Families collection | - Documents      |
| - Password resets   | - Children collection | - Voice recordings |
| - User permissions  | - Activities data   | - Uploaded files   |
|               |               |              |
└──────────────────────────────┴──────────────────────────────┴──────
─────────────────────┘
```
<p><em>Firebase services used in Indaba Care</em></p>
</div>

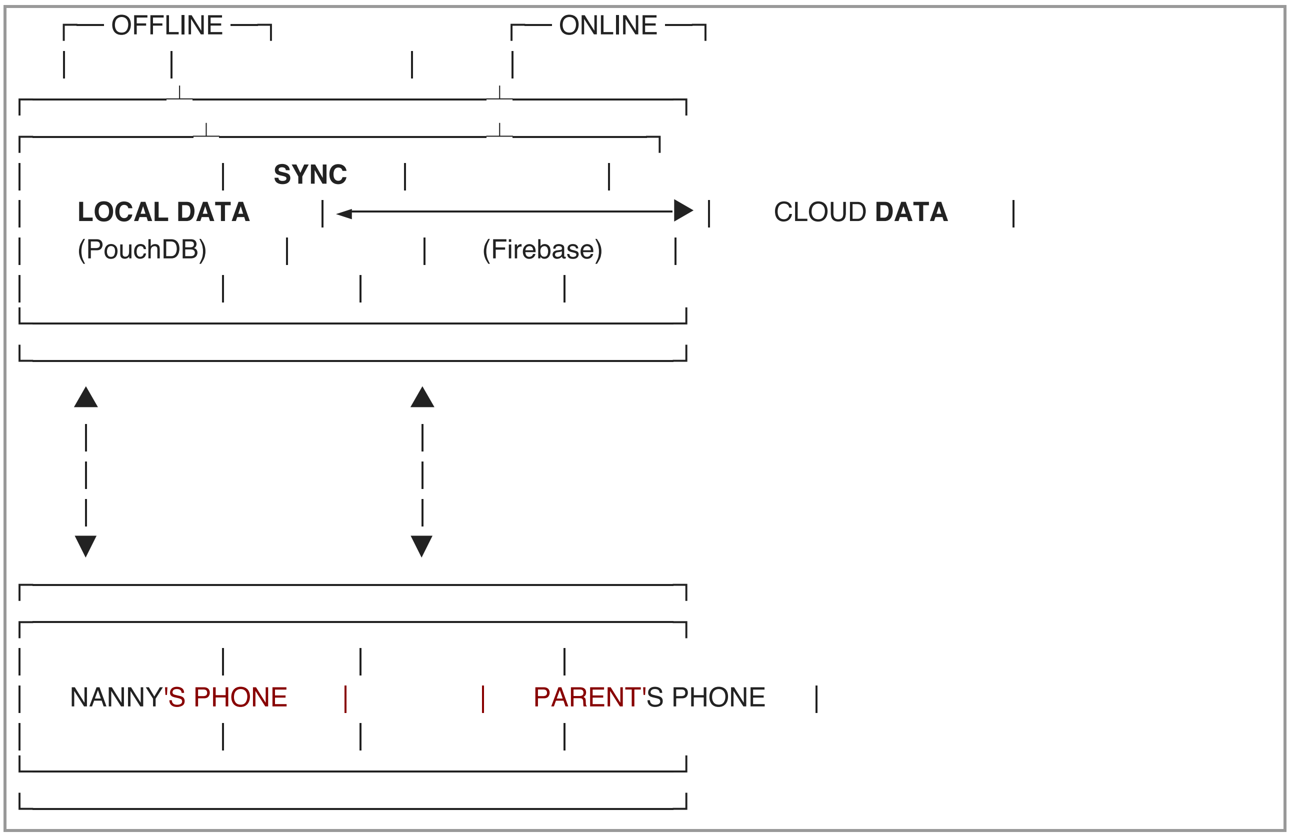
- **Authentication Drawer** (Who are you?)
    - Keeps track of usernames and passwords
    - Verifies people are who they say they are
    - Protects private family information
- **Firestore Database Drawer** (What information do we have?)
    - Holds all the text data - user profiles, family details, children's information
    - Organizes everything in "collections" (like folders) and "documents" (like files)
- **Storage Drawer** (Where are the photos?)
    - Stores photos and larger files
    - Works like a digital photo album

## 2. React & Next.js: The User Interface Builder

This is what creates everything you see on screen - buttons, forms, photos, etc. Think of it as the architect and interior designer for our digital house.

## 3. PouchDB: The Magic Sync Engine

<div align="center">
<img src="https://via.placeholder.com/700x400/F3E5F5/333333" alt="Sync Flow Diagram" width="700"/>



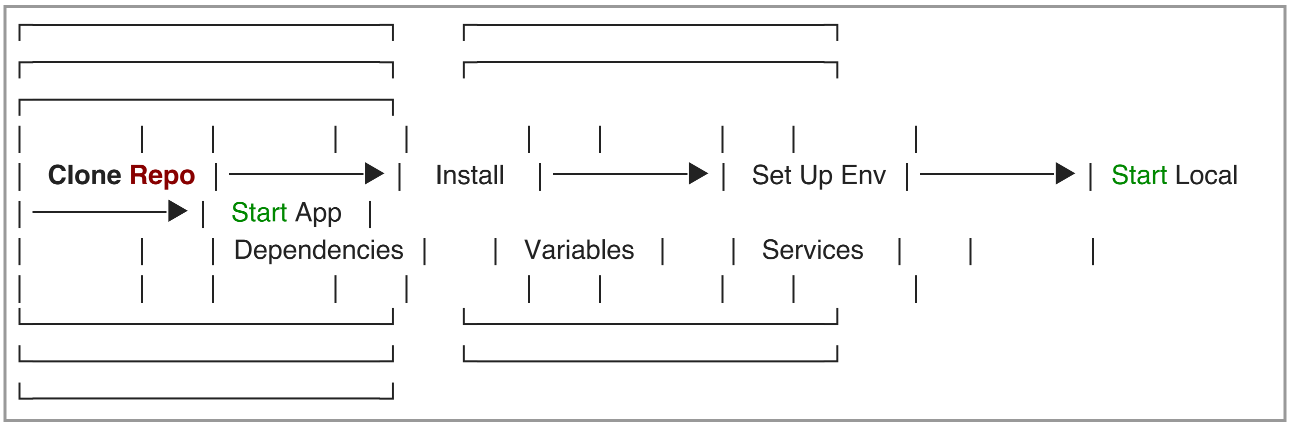
<p><em>How data syncs between devices</em></p>
</div>

This is what makes offline-first work! PouchDB is like a mini-filing cabinet that lives on your phone or computer. It:

- Stores a copy of your data locally
- Quietly syncs with Firebase when internet is available
- Resolves any conflicts if changes were made in multiple places

## Getting Started: Your First Day

<div align="center">
<img src="https://via.placeholder.com/800x400/FFEBEE/333333" alt="Setup Flow" width="800"/>



<p><em>Setup process visualization</em></p>
</div>

### Step 1: Get the Code

```
git clone https://github.com/yourusername/indaba-care.git
cd indaba-care
```

What this does: Downloads the project to your computer, like getting all the blueprints and materials for building a house.

### Step 2: Install the Tools

```
npm install
```

What this does: Gathers all the tools and materials needed to build the app, like gathering hammers, nails, and lumber before constructing a house.

### Step 3: Set Up Your Environment

```
cp .env.demo .env.local
```

What this does: Creates your personal set of "keys" to access the development environment. These keys connect to Firebase services.

### Step 4: Start the Local Development Services

```
# Install global tools
npm install -g firebase-tools

# Set up Firebase locally
firebase login
firebase init emulators

# Start the local services
firebase emulators:start
```

What this does: Creates a miniature version of the entire Firebase filing cabinet on your computer. This means you're not working with real user data, and you can experiment safely.

### Step 5: Start the App

```
npm run dev
```

What this does: Launches the app on your computer at http://localhost:3000 - this address is just your computer talking to itself.

## Meeting the Test Users

<div align="center">
<img src="https://via.placeholder.com/800x300/E8EAF6/333333" alt="Test Users" width="800"/>

```
┌──────────────────────────────────────────┐
┌──────────────────────────────────────────┐
┌──────────────────────────────────────────┐
│                    │    │                    │    │                    │
│      DEMO USER        │    │         PARENT          │    │         NANNY           │
│                    │    │                    │    │                    │
├──────────────────────────────────────────┤
├──────────────────────────────────────────┤
├──────────────────────────────────────────┤
│                    │    │                    │    │                    │
│  demo@indabacare.com    │    │  parent@indabacare.com  │    │  nanny@indabacare.com   │
│  Password: indaba123    │    │  Password: indaba123    │    │  Password: indaba123    │
│                    │    │                    │    │                    │
├──────────────────────────────────────────┤
├──────────────────────────────────────────┤
├──────────────────────────────────────────┤
│                    │    │                    │    │                    │
│  - Full app access      │    │  - Can view all data    │    │  - Limited access       │
│  - Testing account      │    │  - Can add activities   │    │  - Focused on daily     │
│                    │    │  - Manages family       │    │    activities recording │
│                    │    │                    │    │                    │
└──────────────────────────────────────────┘
└──────────────────────────────────────────┘
└──────────────────────────────────────────┘
```

<p><em>Test user accounts for development</em></p>
</div>

We've created pretend people you can log in as:

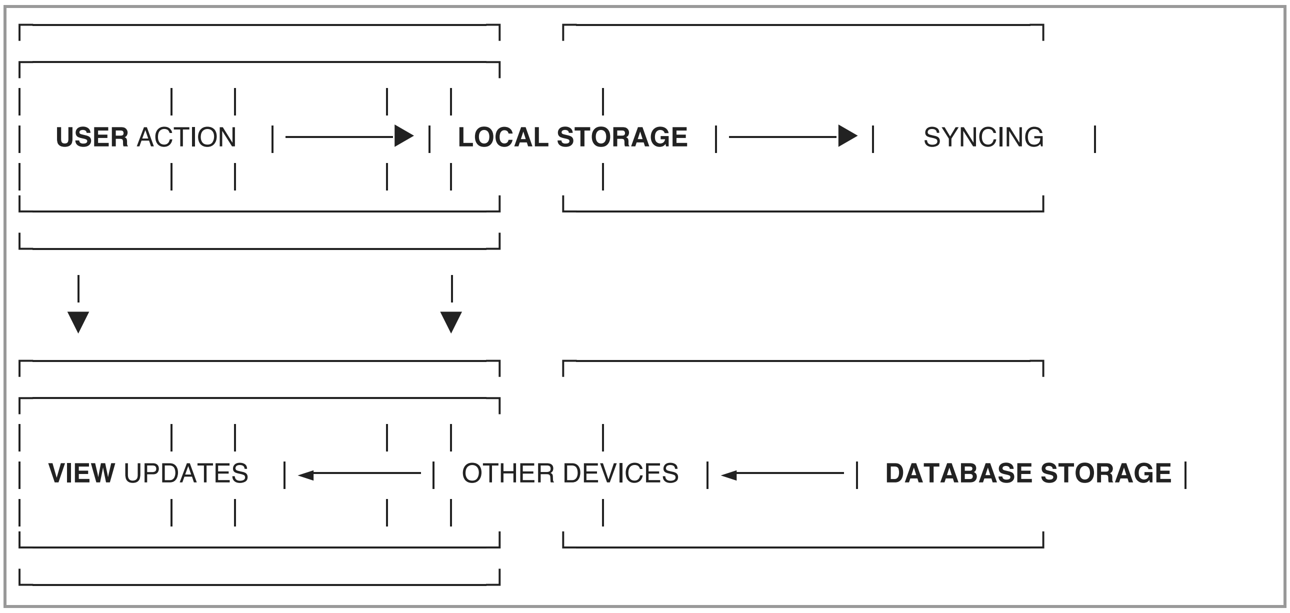
- **Demo User**: demo@indabacare.com / indaba123
- **Parent**: parent@indabacare.com / indaba123
- **Nanny**: nanny@indabacare.com / indaba123

These aren't real people - they're like mannequins we use to test how clothes fit before selling them to real customers.

## Understanding How It All Works Together

### The Data Flow: A Package Delivery Analogy

<div align="center">
<img src="https://via.placeholder.com/800x500/F1F8E9/333333" alt="Data Flow" width="800"/>



<p><em>How data flows through the system</em></p>
</div>

1. **User Action**: A nanny records that a child finished lunch
    - Think of this as: Writing a note about lunch
2. **Local Storage**: The information is saved to PouchDB on their device
    - Think of this as: Putting the note in an envelope
3. **Syncing**: When online, PouchDB syncs with Firebase
    - Think of this as: The postal service collecting the envelope
4. **Database Storage**: Firebase Firestore saves the data
    - Think of this as: Filing the note in the family's folder at the post office
5. **Other Devices**: Other family members' apps sync with Firebase
    - Think of this as: The post office delivering copies to everyone in the family
6. **View Updates**: Everyone's app shows the updated information
    - Think of this as: Family members reading the note

## Security: Digital Locks and Keys

<div align="center">
<img src="https://via.placeholder.com/700x400/FAFAFA/333333" alt="Security Model" width="700"/>

```
┌──────────────────────────────────────────────────────────────
───────────────────────────────────┐
│                                  │
│           AUTHENTICATION                  │
│          (Front Door to the App)        │
│                                  │
└──────────────────────────────────────────────────────────────
───────────────────────────────────┘
                 │
                 ▼
┌──────────────────────────────────────────────────────────────
───────────────────────────────────┐
│                                  │
│          FAMILY-BASED PERMISSIONS              │
│      (**Private** Rooms **for** Each Family's Data)        │
│                                  │
├─────────────────────────────────────────────────┬────────────
───────────────────────────────────┤
│               │                  │
│   PARENT **ACCESS**      │       NANNY **ACCESS**        │
│  (Full Family Cabinet)   │   (Activity Recording Only)    │
│               │                  │
└─────────────────────────────────────────────────┴────────────
───────────────────────────────────┘
```

<p><em>Security model visualization</em></p>
</div>

- **Authentication**: Only verified users can access the app
    - Think of this as: Having a key to the front door
- **Family-Based Permissions**: Users can only see their own family's data
    - Think of this as: Each family having their own private room inside
- **Role-Based Access**: Parents see different things than nannies
    - Think of this as: Different keys opening different cabinets inside the room

## Exploring the Project

### Important Folders and What They Do

<div align="center">
<img src="https://via.placeholder.com/800x500/EFEBE9/333333" alt="Project Structure" width="800"/>

```
indaba-care/
│
├── src/               # All the app's code
│   ├── components/    # Reusable interface parts (like LEGO blocks)
│   ├── pages/         # Each screen in the app
│   ├── contexts/      # Shared information (like the family's calendar)
│   ├── hooks/         # Special functions (like magic spells)
│   └── lib/           # Utility functions (like tools in a toolbox)
│
├── public/            # Images and unchanging files
│
├── tests/             # Code that checks if everything works
│
└── docs/              # Documentation and guides
```

<p><em>Project folder structure simplified</em></p>
</div>

## Making Changes

### Three Levels of Changes

<div align="center">
<img src="https://via.placeholder.com/800x400/E0F2F1/333333" alt="Change Complexity" width="800"/>

```
┌─────────────────────────┐   ┌─────────────────────────┐
┌─────────────────────────┐
│               │   │               │   │               │
│   LEVEL 1       │   │   LEVEL 2       │   │   LEVEL 3       │
│  Simple Changes    │   │  Feature Adjustments │   │ Core Functionality   │
│               │   │               │   │               │
├─────────────────────────┤   ├─────────────────────────┤
├─────────────────────────┤
│               │   │               │   │               │
│  - Text edits      │   │ - Adding fields     │   │ - Sync changes      │
│  - Color changes    │   │  - New screens       │   │  - New integrations  │
│  - Image updates    │   │  - Display changes   │   │  - New features      │
│               │   │               │   │               │
├─────────────────────────┤   ├─────────────────────────┤
├─────────────────────────┤
│               │   │               │   │               │
│  DIFFICULTY: LOW     │   │  DIFFICULTY: MEDIUM  │   │  DIFFICULTY: HIGH    │
│               │   │               │   │               │
└─────────────────────────┘   └─────────────────────────┘
└─────────────────────────┘
```

<p><em>Three levels of code changes by complexity</em></p>
</div>

#### Level 1: Simple Text and Image Changes

- Editing text in files

- Changing colors or images
- Example: Changing "Welcome" to "Hello" on the login page

**Level 2: Feature Adjustments**

- Adding a field to a form
- Creating a new screen using existing patterns
- Changing how information is displayed

**Level 3: Core Functionality Changes**

- Changing how synchronization works
- Adding new Firebase integrations
- Creating entirely new features

## Testing Your Work

### Types of Testing

<div align="center">
<img src="https://via.placeholder.com/700x300/FFF3E0/333333" alt="Testing Types" width="700"/>

```
┌─────────────────────────────────────────────────────────────┐
┌─────────────────────────────────────────────────────────────┐
│                  │       │                  │
│    MANUAL TESTING      │       │    AUTOMATED TESTING     │
│    (Human Testing)     │       │   (Computer Testing)    │
│                  │       │                  │
├─────────────────────────────────────────────────────────────┤
├─────────────────────────────────────────────────────────────┤
│                  │       │                  │
│  ✓ Try all user journeys   │       │  ✓ Run with 'npm test'     │
│  ✓ Test on different devices │       │  ✓ Tests in 'tests/' folder │
│  ✓ Test offline mode      │       │  ✓ Runs automatically     │
│  ✓ Look for visual issues   │       │  ✓ Finds code problems     │
│                  │       │                  │
└─────────────────────────────────────────────────────────────┘
└─────────────────────────────────────────────────────────────┘
```

<p><em>Two complementary approaches to testing</em></p>
</div>

1. **Manual Testing**: Use the app yourself
   - Log in with test accounts
   - Try the feature you changed
   - Check different devices (phone, tablet, computer)
2. **Automated Testing**: Let the computer check
   - Run npm test to check everything
   - Tests are in the tests/ folder

# The Special Challenges of Offline-First

<div align="center">
<img src="https://via.placeholder.com/800x400/F9FBE7/333333" alt="Offline Challenges" width="800"/>

```
┌─────────────────────────────────────────────────────────────────────────
─────────────────────────────────────────┐
│                 OFFLINE-FIRST CHALLENGES                     │
├──────────────────────────────────────────────────────┬──────────────────
──────────────────────────────────────────┤
│                    │                               │
│  CONFLICTING CHANGES       │           PHOTO STORAGE            │
│                    │                               │
├──────────────────────────────────────────────────────┼──────────────────
──────────────────────────────────────────┤
│                    │                               │
│  Problem:                │  Problem:                        │
│  Multiple offline changes  │  Photos are large files                │
│  to the same data          │  Hard to store many offline            │
│                    │                               │
│  Solution:               │  Solution:                       │
│  - Timestamps              │  - Compress photos                  │
│  - Conflict resolution     │  - Upload queue                     │
│  - Last write wins         │  - Placeholders until uploaded        │
│                    │                               │
└──────────────────────────────────────────────────────┴──────────────────
──────────────────────────────────────────┘
```

<p><em>Key challenges in building offline-first applications</em></p>
</div>

### Challenge 1: Conflicting Changes

What happens if a parent and nanny both change something while offline?

- **Example**: Parent marks medicine as given at 2pm while offline
- **Also**: Nanny marks the same medicine as given at 2:15pm while offline
- **Result**: When both sync, the system must decide which is correct

Our solution uses timestamps and conflict resolution rules.

### Challenge 2: Photo Storage

Photos are large and tricky to handle offline:

- We compress them on the device first
- They're queued for upload when back online
- A placeholder shows until upload completes

## When Things Go Wrong

<div align="center">
<img src="https://via.placeholder.com/700x400/FFEBEE/333333" alt="Troubleshooting" width="700"/>

```
┌─────────────────────────────────────────────────────────────────────
─────────────────────────────────────┐
│                 COMMON PROBLEMS                        │
├────────────────────────────────────────────────────────┬────────────
─────────────────────────────────────┤
│                        │                          │
│  "Cannot connect to Firebase" │       Login not working         │
│                        │                          │
├────────────────────────────────────────────────────────┼────────────
─────────────────────────────────────┤
│  ✓ Check emulators running  │  ✓ Check exact credentials       │
│  ✓ Check .env.local config  │  ✓ Check Auth emulator running   │
│                        │                          │
├────────────────────────────────────────────────────────┼────────────
─────────────────────────────────────┤
│                        │                          │
│  Changes not showing up     │       Tests failing              │
│                        │                          │
├────────────────────────────────────────────────────────┼────────────
─────────────────────────────────────┤
│  ✓ Refresh the page         │  ✓ Read error message            │
│  ✓ Check console for errors │  ✓ Check what changed recently   │
│                        │                          │
└────────────────────────────────────────────────────────┴────────────
─────────────────────────────────────┘
```

<p><em>Common problems and solutions</em></p>
</div>

### Getting Help

- Check documentation in the docs/ folder
- Look for similar issues in the code
- Ask in the team Slack channel

## Learning More

The best way to learn is to explore! Try:

1. **Reading the code**: Even if you don't understand everything, patterns will emerge
2. **Making small changes**: See what happens when you alter things
3. **Following the data**: Pick one piece of information and follow it through the system

Remember: Everyone was a beginner once. The more you explore, the more you'll understand!

## Complete Setup Guide: From Zero to Running App

<div align="center">
<img src="https://via.placeholder.com/800x200/E1F5FE/333333?text=Complete+Setup+Journey" alt="Setup Journey" width="800"/>

```
┌─────────────────┐   ┌─────────────────┐   ┌─────────────────┐
┌─────────────────┐   ┌─────────────────┐
│         │   │         │   │         │   │         │   │         │
│ Firebase  │───▶│ Environment │───▶│ Emulators │───▶│ Next.js  │───▶│ Testing  │
│  Setup   │   │  Setup   │   │  Setup   │   │  Setup   │   │         │
│         │   │         │   │         │   │         │   │         │
└─────────────────┘   └─────────────────┘   └─────────────────┘
└─────────────────┘   └─────────────────┘
```

<p><em>The complete setup journey</em></p>
</div>

## 1. Firebase Project Setup

<div align="center">
<img src="https://via.placeholder.com/800x400/E3F2FD/333333" alt="Firebase Setup" width="800"/>

```
┌─────────────────────────────────────────────────────────────────
─────────────────────────────┐
│                FIREBASE SETUP STEPS                   │
├─────────────────────────────────────────────────────────────────
─────────────────────────────┤
│                                                              │
│  1. CREATE FIREBASE PROJECT                                  │
│    - Go to Firebase Console (firebase.google.com)            │
│    - Click "Add project"                                     │
│    - Name it "Indaba Care"                                   │
│    - Enable Google Analytics                                 │
│                                                              │
├─────────────────────────────────────────────────────────────────
─────────────────────────────┤
│                                                              │
│  2. REGISTER WEB APP                                         │
│    - In Project Overview, click web icon </>                 │
│    - Name it "Indaba Care Web"                               │
│    - Copy the firebaseConfig values                          │
│                                                              │
├─────────────────────────────────────────────────────────────────
─────────────────────────────┤
│                                                              │
│  3. ENABLE AUTHENTICATION                                    │
│    - Go to Authentication > Get Started                      │
│    - Enable Email/Password                                   │
│    - Enable Google Sign-in                                   │
│                                                              │
├─────────────────────────────────────────────────────────────────
─────────────────────────────┤
│                                                              │
│  4. SET UP FIRESTORE DATABASE                                │
│    - Go to Firestore Database > Create database              │
│    - Start in production mode                                │
│    - Choose region: europe-west1 (Belgium)                   │
│                                                              │
├─────────────────────────────────────────────────────────────────
─────────────────────────────┤
│                                                              │
│  5. SET UP FIREBASE STORAGE                                  │
│    - Go to Storage > Get Started                             │
│    - Start in production mode                                │
│    - Use Firebase emulator for development                   │
│                                                              │
└─────────────────────────────────────────────────────────────────
─────────────────────────────┘
```

<p><em>Step-by-step Firebase setup process</em></p>
</div>

**Create Test User Accounts**

```
# Start Firebase emulators first
firebase emulators:start

# In another terminal, use Firebase CLI to create test users
firebase auth:import test-users.json
```

Where test-users.json contains:

```
{
  "users": [
    {
      "localId": "demo123",
      "email": "demo@indabacare.com",
      "passwordHash": "...",
      "displayName": "Demo User"
    },
    {
      "localId": "parent123",
      "email": "parent@indabacare.com",
      "passwordHash": "...",
      "displayName": "Test Parent"
    },
    {
      "localId": "nanny123",
      "email": "nanny@indabacare.com",
      "passwordHash": "...",
      "displayName": "Test Nanny"
    }
  ]
}
```

## 2. Environment Configuration

<div align="center">
<img src="https://via.placeholder.com/800x300/E8F5E9/333333" alt="Environment Setup" width="800"/>

```
┌──────────────────────────────────────────────────────────────
───────────────────────┐
│              ENVIRONMENT SETUP STEPS                   │
├──────────────────────────────────────────────────────────────
──────────────────────┤
│                                                       │
│  1. CREATE .env.local FILE                             │
│     - Copy .env.demo to .env.local                     │
│     - This file contains Firebase configuration        │
│                                                       │
├──────────────────────────────────────────────────────────────
──────────────────────┤
│                                                       │
│  2. IMPORTANT VALUES IN .env.local                     │
│     - NEXT_PUBLIC_FIREBASE_API_KEY                     │
│     - NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN                 │
│     - NEXT_PUBLIC_FIREBASE_PROJECT_ID                  │
│     - NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET              │
│     - NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID         │
│     - NEXT_PUBLIC_FIREBASE_APP_ID                      │
│                                                       │
├──────────────────────────────────────────────────────────────
──────────────────────┤
│                                                       │
│  3. EMULATOR SETTINGS IN .env.local                    │
│     - FIRESTORE_EMULATOR_HOST=localhost:8080           │
│     - FIREBASE_AUTH_EMULATOR_HOST=localhost:9099       │
│     - FIREBASE_STORAGE_EMULATOR_HOST=localhost:9199    │
│                                                       │
└──────────────────────────────────────────────────────────────
───────────────────────┘
```

<p><em>Environment setup process</em></p>
</div>

### 3. Firebase Emulators Setup

<div align="center">
<img src="https://via.placeholder.com/800x500/FFF8E1/333333" alt="Emulator Setup" width="800"/>

```
┌─────────────────────────────────────────────────────────────
─────────────────────────────┐
│              FIREBASE EMULATORS EXPLAINED                │
├─────────────────────────────────────────────────────────────
─────────────────────────────┤
│                                                          │
│  WHAT ARE EMULATORS?                                     │
│  - Local versions of Firebase services                   │
│  - Run on your computer, not in the cloud                │
│  - Let you develop without using actual Firebase resources   │
│  - Free to use and won't affect production data          │
│                                                          │
├─────────────────────────────────────────────────────────────
────────────────┬────────────┤
│                                                │  PORT   │
├─────────────────────────────────────────────────────────────
────────────────┼────────────┤
│  1. FIRESTORE EMULATOR                         │  8080   │
│     - Local database for development           │         │
│                                                │         │
├─────────────────────────────────────────────────────────────
────────────────┼────────────┤
│  2. AUTHENTICATION EMULATOR                    │  9099   │
│     - Handles user login/signup locally        │         │
│                                                │         │
├─────────────────────────────────────────────────────────────
────────────────┼────────────┤
│  3. STORAGE EMULATOR                           │  9199   │
│     - Stores photos and files locally          │         │
│                                                │         │
└─────────────────────────────────────────────────────────────
────────────────┴────────────┘
```

<p><em>Firebase Emulators explained</em></p>
</div>

**Setting Up Firebase Emulators**

```
# Install Firebase tools globally
npm install -g firebase-tools

# Log in to Firebase
firebase login

# Initialize Firebase in your project
firebase init
# Select: Firestore, Storage, Emulators
# For emulators, select: Auth, Firestore, Storage

# Start all emulators
firebase emulators:start
```

The emulator UI will be available at http://localhost:4000

**Firebase Security Rules**

For Firestore, create a file firestore.rules:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    // Allow authenticated users to read their own data
    match /users/{userId} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }

    // Allow family members to access shared family data
    match /families/{familyId} {
      allow read, write: if request.auth != null &&
        exists(/databases/$(database)/documents/families/$(familyId)/members/$(request.auth.uid));
    }

    // Allow access to children data by family members
    match /children/{childId} {
      allow read, write: if request.auth != null &&
        exists(/databases/$(database)/documents/children/$(childId)/familyId) &&

exists(/databases/$(database)/documents/families/get(/databases/$(database)/documents/children/$(childId)
/familyId).data.id/members/$(request.auth.uid));
    }
  }
}
```

For Storage, create a file storage.rules:

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    // Allow authenticated users to read/write their own files
    match /users/{userId}/{allPaths=**} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }

    // Allow family members to access shared family photos
    match /families/{familyId}/{allPaths=**} {
      allow read, write: if request.auth != null &&
        exists(/databases/$(database)/documents/families/$(familyId)/members/$(request.auth.uid));
    }
  }
}
```

## 4. Sample Data Creation

<div align="center">
<img src="https://via.placeholder.com/800x400/F3E5F5/333333" alt="Sample Data Structure" width="800"/>

```
┌──────────────────────────────────────────────────────────────────────────
────────────────────────────────┐
│                SAMPLE DATA STRUCTURE                    │
├─────────────────────────────┬────────────────────────────┬──────
─────────────────────────────┤
│              │               │               │
│  USERS COLLECTION  │  FAMILIES COLLECTION  │  CHILDREN COLLECTION  │
│              │               │               │
├─────────────────────────────┼────────────────────────────┼──────
─────────────────────────────┤
│              │               │               │
│  - userId          │  - familyId          │  - childId           │
│  - email           │  - name              │  - name              │
│  - displayName     │  - address           │  - dateOfBirth       │
│  - role            │  - members: [userIds] │  - familyId          │
│  - photoURL        │  - createdAt         │  - allergies         │
│  - familyIds       │  - updatedAt         │  - medications       │
│              │               │  - createdAt         │
│              │               │               │
└─────────────────────────────┴────────────────────────────┴──────
─────────────────────────────┘
```

<p><em>Key collections and their fields</em></p>
</div>

Create a script to populate your emulator with test data:

```
// scripts/seed-data.js
const admin = require('firebase-admin');
const serviceAccount = require('./service-account-key.json');

admin.initializeApp({
  credential: admin.credential.cert(serviceAccount)
});

const db = admin.firestore();

// Sample user data
const users = [
  {
    uid: 'demo123',
    email: 'demo@indabacare.com',
    displayName: 'Demo User',
    role: 'admin',
    familyIds: ['family1']
  },
  // Add more users...
];
```

```
// Sample family data
const families = [
  {
    id: 'family1',
    name: 'Johnson Family',
    address: '123 Main St, Cape Town',
    members: ['demo123', 'parent123', 'nanny123'],
    createdAt: admin.firestore.FieldValue.serverTimestamp(),
    updatedAt: admin.firestore.FieldValue.serverTimestamp()
  },
  // Add more families...
];

// Sample children data
const children = [
  {
    id: 'child1',
    name: 'Sam Johnson',
    dateOfBirth: new Date('2020-05-15'),
    familyId: 'family1',
    allergies: ['nuts', 'dairy'],
    medications: [],
    createdAt: admin.firestore.FieldValue.serverTimestamp()
  },
  // Add more children...
];

// Insert sample data
async function seedData() {
  // Add users
  for (const user of users) {
    await db.collection('users').doc(user.uid).set(user);
  }

  // Add families
  for (const family of families) {
    await db.collection('families').doc(family.id).set(family);
  }

  // Add children
  for (const child of children) {
    await db.collection('children').doc(child.id).set(child);
  }

  console.log('Sample data has been added!');
}

seedData();
```

## 5. Running and Testing the Application

<div align="center">
<img src="https://via.placeholder.com/800x400/E0F7FA/333333" alt="Testing Flow" width="800"/>

```
┌──────────────────────────────────────────────────────┐
│                APPLICATION TESTING                   │
├──────────────────────────────────────────────────────┤
│                                                      │
│  1. START FIREBASE EMULATORS                         │
│     $ firebase emulators:start                       │
│                                                      │
├──────────────────────────────────────────────────────┤
│                                                      │
│  2. START DEVELOPMENT SERVER                         │
│     $ npm run dev                                    │
│                                                      │
├──────────────────────────────────────────────────────┤
│                                                      │
│  3. OPEN BROWSER                                     │
│     http://localhost:3000                            │
│                                                      │
├──────────────────────────────────────────────────────┤
│                                                      │
│  4. TEST USER JOURNEYS                               │
│     - Login with test users                          │
│     - Add activities                                 │
│     - Test offline functionality                     │
│     - Upload photos                                  │
│                                                      │
└──────────────────────────────────────────────────────┘
```

<p><em>Application testing workflow</em></p>
</div>

## Preparing for Production

<div align="center">
<img src="https://via.placeholder.com/800x400/F1F8E9/333333" alt="Production Setup" width="800"/>

```
┌──────────────────────────────────────────────────────────────────
──────────────────────────────┐
│          MOVING FROM DEVELOPMENT TO PRODUCTION           │
├──────────────────────────────────────────────┬──────────────────
──────────────────────────────┤
│                    │                        │
│  DEVELOPMENT ENVIRONMENT    │       PRODUCTION ENVIRONMENT      │
│                    │                        │
├──────────────────────────────────────────────┼──────────────────
──────────────────────────────┤
│                    │                        │
│  - Local emulators         │  - Real Firebase services        │
│  - Test user accounts       │  - Real user accounts            │
│  - Sample data             │  - Real data                     │
│  - Lenient security rules   │  - Strict security rules         │
│  - Environment: .env.local  │  - Environment: production vars   │
│                    │                        │
└──────────────────────────────────────────────┴──────────────────
──────────────────────────────┘
```

<p><em>Key differences between development and production</em></p>
</div>

## Steps for Production Deployment

1. **Update Firebase Security Rules**
    - Review and tighten security rules for Firestore and Storage
    - Deploy updated rules to production: firebase deploy --only firestore:rules,storage:rules
2. **Set Up Authentication Providers**
    - Configure proper redirect URLs for Google Sign-in
    - Add domain to authorized domains list
3. **Prepare Environment Variables**
    - Create production-specific environment variables
    - Never use emulator settings in production
4. **Deploy the Application**
    - Build the production version: npm run build
    - Deploy to hosting: firebase deploy --only hosting
    - Alternatively, deploy to Vercel or Netlify
5. **Set Up Analytics and Monitoring**
    - Enable Firebase Analytics
    - Configure Crash Reporting
    - Set up Performance Monitoring

## Keeping Our App Safe: Security Made Simple

Imagine our app is like a house where families store precious photos and private information. Just as we need locks on doors and windows to keep a house safe, our app needs security measures to protect family data.

<div align="center">
<img src="https://via.placeholder.com/800x400/FFEBEE/333333" alt="Security Explained" width="800"/>

```
┌─────────────────────────────────────────────────────────────────────────
────────────────────────────────────┐
│                 THE DIGITAL HOUSE                    │
│                                             │
│                 ┌──────────────────────────┐                    │
│                 │  FRONT    │                   │
│                 │  **DOOR**    │  ← Authentication:        │
│                 │        │    Who can enter?        │
│                 └──────────────────────────┘                    │
│                    │                        │
│                    ▼                        │
│    ┌───────────────────────────────────┐   ┌──────────────────────────┐
┌────────────────────────────────────┐     │
│    │           │    │         │    │           │      │
│    │  PARENT'S    │    │  SHARED    │    │  NANNY'S     │      │
│    │  **ROOM**       │    │  LIVING    │    │  **ROOM**       │      │
│    │           │    │  **ROOM**    │    │           │      │
│    └───────────────────────────────────┘   └──────────────────────────┘
└────────────────────────────────────┘     │
│                    │                        │
│                    │                        │
│                    ▼                        │
│                ┌────────────────────────────┐                    │
│                │  VALUABLE   │  ← Rules protect          │
│                │  CABINET   │    sensitive data        │
│                └────────────────────────────┘                    │
│                                             │
└─────────────────────────────────────────────────────────────────────────
────────────────────────────────────┘
```

<p><em>How security works in our app</em></p>
</div>

## 1. The Front Door: User Accounts & Passwords

**What it is:** Just like your front door keeps strangers out of your house, user accounts and passwords prevent unauthorized people from accessing the app.

**Why it matters:** Without good locks (passwords), anyone could access private family information.

**How it works in everyday terms:**

<div align="center">
<img src="https://via.placeholder.com/800x300/FFF3E0/333333" alt="Authentication Explained" width="800"/>

```
┌─────────────────────────────┐      ┌─────────────────────────────┐
┌─────────────────────────────┐
|             |         |             |        |             |
|   WEAK LOCK     |           |   BETTER LOCK    |          |    BEST LOCK    |
|             |         |             |        |             |
├─────────────────────────────┤      ├─────────────────────────────┤
├─────────────────────────────┤
|             |         |             |        |             |
|  password123    |           |  J7h!K9$p       |           |  J7h!K9$p       |
|             |         |             |        |    +        |
|             |         |             |        |  Phone Code     |
|             |         |             |        |             |
├─────────────────────────────┤      ├─────────────────────────────┤
├─────────────────────────────┤
|             |         |             |        |             |
|  Easy to guess   |           |  Hard to guess   |           |  Two different   |
|  Like a simple   |           |  Like a complex  |           |  keys needed     |
|  doorknob lock   |           |  deadbolt        |           |  (2-factor)      |
|             |         |             |        |             |
└─────────────────────────────┘      └─────────────────────────────┘
└─────────────────────────────┘
```

<p><em>Different types of password security</em></p>
</div>

**How you can improve it:**

- Ask users to create passwords with at least 8 characters including numbers and symbols
- Turn on email verification so people prove they own the email they signed up with
- For administrator accounts, require two-factor authentication (something they know AND something they have)

## 2. Room Access: Who Can See What?

**What it is:** Just like different rooms in a house can have different access rules (everyone can use the living room, but bedrooms are private), our app has rules about who can see and change what data.

**Why it matters:** Parents might need to see everything about their child's care, but a nanny should only see information relevant to their work.

**How it works in everyday terms:**

<div align="center">
<img src="https://via.placeholder.com/800x400/E8F5E9/333333" alt="Data Access Rules" width="800"/>

```
┌──────────────────────────────────────────────────────────────────────────
──────────────────────────────┐
│              ROOM ACCESS PERMISSIONS                │
├────────────────────────────────┬─────────────────────────────────┬───────
──────────────────────────┐
│              │              │              │
│    PARENT      │     NANNY      │   VISITOR      │
│              │              │              │
├────────────────────────────────┼─────────────────────────────────┼───────
──────────────────────────┤
│              │              │              │
│ ✓ Child's Room     │ ✓ Child's Room     │ ✗ Child's Room     │
│ ✓ Activity Records │ ✓ Activity Records │ ✗ Activity Records │
│ ✓ Medical Records  │ ✓ Medical Alerts   │ ✗ Medical Records  │
│ ✓ Family Calendar  │ ✓ Work Schedule    │ ✗ Family Calendar  │
│ ✓ Payment Details  │ ✗ Payment Details  │ ✗ Payment Details  │
│ ✓ Add Other Users  │ ✗ Add Other Users  │ ✗ Add Other Users  │
│              │              │              │
└────────────────────────────────┴─────────────────────────────────┴───────
──────────────────────────┘
```

<p><em>Different users have different access levels</em></p>
</div>

**How you can improve it:**

- Give each user only the minimum access they need to do their job
- Check data before saving it (e.g., don't allow obviously wrong birthdates)
- Create clear rules for who can share photos and information

## 3. The Key to Our House: API Keys & Why They Matter

**What it is:** An API key is like a special key that lets our app talk to Firebase (our digital storage space). It's necessary, but we need to be careful with it.

**Why it matters:** If someone steals this key, they could potentially access our Firebase project.

**How it works in everyday terms:**

<div align="center">
<img src="https://via.placeholder.com/800x350/E3F2FD/333333" alt="API Key Security" width="800"/>

```
┌──────────────────────────────────────────────────────────────────────────
──────────────────────────────────┐
|                  THE DIGITAL KEY                          |
├──────────────────────────────────────────────────────────────────────────
──────────────────────────────────┤
|                                                           |
|  NEXT_PUBLIC_FIREBASE_API_KEY=AIzaSyBUD1-oNapkrHk71Lbo0...      |
|                                                           |
├──────────────────────────────────────────────────────────────────────────
──────────────────────────────────┤
|                                                           |
|  Why we need it:     ↔  To connect our app to Firebase          |
|                                                           |
|  Is it secret?       ↔  Not completely (it's in browser code)    |
|                                                           |
|  Security strategy:  ↔  Restrict what this key can do           |
|                                                           |
└──────────────────────────────────────────────────────────────────────────
──────────────────────────────────┘
```

<p><em>Understanding the Firebase API key</em></p>
</div>

**Important:** Firebase API keys are different from most API keys. They appear in the browser and aren't completely secret, but we can make them safer by restricting where they can be used.

**How to secure your Firebase API key:**

1. **Add a website restriction:** This is like saying "this key only works at this address."

   <div align="center">
   <img src="https://via.placeholder.com/700x300/F9FBE7/333333" alt="API Key Restriction" width="700"/>

---

```
┌──────────────────────────────────────────────────────────────────────────
| RESTRICTING YOUR API KEY  |
├──────────────────────────────────────────────────────────────────────────
|  |
| 1. Go to Firebase Console  |
| 2. Click the gear icon ⚙ for "Project settings"  |
| 3. Go to the "API keys" tab  |
| 4. Find your Web API key  |
| 5. Click "Application restrictions"  |
| 6. Select "HTTP referrers"  |
| 7. Add your website domain (e.g., indabacare.com)  |
| 8. Click "Save"  |
|  |
└──────────────────────────────────────────────────────────────────────────
```

```
<p><em>Steps to restrict API key usage</em></p>
</div>

2. **Set up Firebase Security Rules:** These are like instructions to the storage room: "Only let people see
what they're allowed to see."

### 4. Teaching Users About Security

Just like we teach children about home safety, it's important to teach app users about digital safety:

* Show family members how to create strong passwords
* Explain to nannies why they should never share their login details
* Create clear guidelines about sharing photos of children
* Include simple security tips in the app

<div align="center">
<img src="https://via.placeholder.com/800x300/E0F7FA/333333" alt="User Education" width="800"/>
```

```
┌──────────────────────────────────────────────────────────────────
│ SECURITY EDUCATION │
├──────────────────────────────────────────────────────────────────
│ │
│ • Use unique passwords for each app or website │
│ • Never share your password with others │
│ • Be careful about what photos you upload │
│ • Log out when using shared computers │
│ • Watch out for suspicious emails asking for your password │
│ • Use the "forgot password" feature if something seems wrong │
│ │
└──────────────────────────────────────────────────────────────────
```

```
 <p><em>Security tips for users</em></p>
</div>

### 5. When Things Feel Unsafe: Security Checklist

If you ever worry the app might have security issues, go through this checklist:

<div align="center">
 <img src="https://via.placeholder.com/800x400/F3E5F5/333333" alt="Security Checklist" width="800"/>
```

```
┌──────────────────────────────────────────────────────────────────
│ SECURITY CHECKLIST │
├───────────────────────────────────────────────┬──────────────────
│ │ │
│ QUESTION │ WHAT TO CHECK │
│ │ │
├───────────────────────────────────────────────┼──────────────────
│ │ │
│ Is our data visible to the wrong │ Review Firestore and │
```

| | |
|---|---|
| people? | Storage security rules |
| Are account passwords strong enough? | Check password policy settings |
| Could data be accidentally deleted? | Check backup settings and who has delete access |
| Is our API key restricted? | Check API key settings in Firebase Console |

```
  <p><em>Checklist for reviewing security</em></p>
</div>

##      Current Challenges & Immediate Next Steps

<div align="center">
  <img src="https://via.placeholder.com/800x400/FFE0B2/333333" alt="Current Challenges" width="800"/>
```

| CURRENT STATUS & CHALLENGES | |
|---|---|
| WORKING PROPERLY | CURRENT CHALLENGES |
| ✓ Firebase Authentication | ✗ Firebase Storage (Cloud) |
| ✓ Firestore Database | ✗ Region compatibility issues |
| ✓ Local development setup | ✗ Production deployment |
| ✓ Firebase emulators | |
| ✓ Test user accounts | |

```
  <p><em>Current status and challenges</em></p>
</div>

### The Firebase Storage Challenge Explained

**What's the issue?** We currently can't set up Firebase Storage in the cloud for our production
environment due to region compatibility issues. This affects our ability to store and share photos and
documents.

<div align="center">
  <img src="https://via.placeholder.com/800x400/FFECB3/333333" alt="Storage Challenge" width="800"/>
```

```
┌──────────────────────────────────────────────
|  STORAGE CHALLENGE  |
├──────────────────────────────────────────────
|   |
|  PROBLEM:  |
|  We chose europe-west1 (Belgium) for our Firestore database to  |
|  optimize for South African users, but the free tier of Firebase  |
|  Storage doesn't allow us to select this region.  |
|   |
├──────────────────────────────────────────────
|   |
|  TEMPORARY SOLUTION:  |
|  We're using the Firebase Storage emulator for development.  |
|  This allows us to build and test photo upload/sharing features  |
|  without an actual cloud storage bucket.  |
|   |
├──────────────────────────────────────────────
|   |
|  LIMITATION:  |
|  The emulator doesn't persist data between restarts and isn't  |
|  accessible outside your local computer.  |
|   |
└──────────────────────────────────────────────
```

```
  <p><em>Understanding the storage challenge</em></p>
</div>

### Immediate Next Steps

<div align="center">
  <img src="https://via.placeholder.com/800x500/E3F2FD/333333" alt="Next Steps" width="800"/>
```

```
┌──────────────────────────────────────────────
|  NEXT STEPS  |
├──────────────────────────────────────────────
|   |
|  SHORT-TERM (1-2 WEEKS):  |
|  1. Continue using Storage emulator for development  |
| 2. Test photo upload feature with emulator |
| 3. Update documentation to clarify emulator usage |
| 4. Ensure junior developers can set up emulators correctly |
| |
├──────────────────────────────────────────────────────────────
| |
| MID-TERM (2-4 WEEKS): |
| 1. Explore solutions for cloud Storage: |
| a. Upgrade to paid Firebase plan for region selection |
| b. Use US-CENTRAL1 region despite higher latency |
| c. Explore alternative storage solutions |
| 2. Implement Storage solution based on decision |
| |
├──────────────────────────────────────────────────────────────
| |
| LONG-TERM (1-3 MONTHS): |
| 1. Prepare for production deployment |
| 2. Implement comprehensive backup strategy |
| 3. Set up monitoring for storage usage and costs |
| 4. Create disaster recovery plan |
| |
└──────────────────────────────────────────────────────────────
```

```
  <p><em>Action plan for the coming weeks</em></p>
</div>

### Storage Options Comparison

<div align="center">
  <img src="https://via.placeholder.com/800x400/F1F8E9/333333" alt="Storage Options" width="800"/>
```

| FIREBASE STORAGE<br>(US-CENTRAL1) | REGIONAL SOLUTION<br>(PAID TIER) | ALTERNATIVE STORAGE<br>(AWS S3, etc.) |
|---|---|---|
| + Free to use | + Optimal latency for SA | + Full regional control |
| + Easy integration | + Same region as Firestore | + Potentially lower cost |
| + Works immediately | + Better user experience | + More storage features |
| - Higher latency for SA | - Monthly cost ($25-50) | - More complex setup |
| - Not ideal performance | - Requires billing setup | - Custom integration |
| | | - Additional learning |

<p><em>Comparing **storage options**</em></p>
</div>

## Mobile App Integration

Indaba Care **is** designed **to work on both** web **and** mobile platforms. The mobile app will be built **using** React Native, sharing much **of** the same code **with** the web **version**.

<div align="center">
<img src="https://via.placeholder.com/800x400/E8EAF6/333333" alt="Mobile Integration" width="800"/>

```
┌──────────────────────────────────────────────────────────────
| SHARED ARCHITECTURE |
├─────────────────────────────────────────────┬────────────────
| | |
| WEB APPLICATION | MOBILE APPLICATION |
| (React + Next.js) | (React Native) |
| | |
├─────────────────────────────────────────────┴────────────────
| |
| SHARED BUSINESS LOGIC |
| |
├──────────────────────────────────────────────────────────────
| |
| FIREBASE SERVICES |
| Authentication | Firestore Database | Storage |
| |
└──────────────────────────────────────────────────────────────
```

<p><em>Shared architecture between web and mobile</em></p>
</div>

---

<div align="center">
<p>Created by the Indaba Care Development Team | Last Updated: May 13, 2025</p>
</div>