# Interactive Form Validation

## Phase 2 — Solution Design & Architecture

By
Aruna M – 2023103079
Harsika V – 2023103588
Sharan Saminathan – 2023103609
Sri Bavan Akash S – 2023103627
Paril T – 2023103714

# Tech Stack Selection

Recommended technologies and rationale:

• HTML5 + CSS3 (Tailwind CSS recommended) — Fast prototyping, responsive utilities, consistent UI states.

• JavaScript (ES6+) — Core application logic and validators; works in all modern browsers.

• React (optional; recommended) — Component model and hooks simplify state, effects, and testing. Use vanilla JS if frameworks are restricted.

• Vite — Fast dev server and build tool for quick iteration.

• Jest + Testing Library — Unit and integration testing for validators and components.

• TypeScript (optional) — Static types for validator inputs/outputs to reduce runtime bugs.

• Accessibility tools (axe, Lighthouse) — Automated checks to ensure WCAG compliance.

• Mock server (json-server) or in-app mocks — For username uniqueness checks and submit endpoint during demo.

• Deployment: GitHub Pages / Netlify / Vercel — Simple static site hosts for demos.

# UI Structure & API Schema Design

UI Structure (Pages / Components):

• Registration (primary demo)

• Contact (secondary demo)

• Demo / Accessibility (show live-region and screen-reader behaviors)

• Top-level components: App, FormContainer, InputField, PasswordField, PhoneInput, DateInput, FormSummary, Toast

API Schema (mock endpoints):

• POST /api/validate-username

 Request: { "username": "string" }

 Response: { "available": true | false, "suggestions"?: ["string"] }

• POST /api/submit-form

Request: Full form payload (see sample below)

Response: { "success": true } or { "success": false, "errors": { ... } }

Sample registration payload:

{ "fullName": "Ananya Sharma", "email": "ananya.sharma@example.com", "username": "ananya_s", "password": "********", "phone": "+918765432109", "dob": "1999-07-22"}

# Data Handling Approach

State & data flow:

• Local field state (useState) for individual inputs; Form-level state (useReducer or context) for aggregated validity.

• ValidationResult shape: { valid: boolean, errors: string[], warnings?: string[] }

Synchronous vs Asynchronous validation:

• Synchronous: regex and length checks — run on input and blur.

• Asynchronous: remote uniqueness checks (debounced, 500ms) with cancellation of in-flight requests.

Sanitization & security:

• Trim inputs on blur and before submit.

• Normalize phone numbers to E.164-like format for storage.

• Never persist raw passwords to localStorage; keep them ephemeral in memory.

• Escape/encode displayed user data to prevent XSS in demo outputs.

# Component / Module Diagram
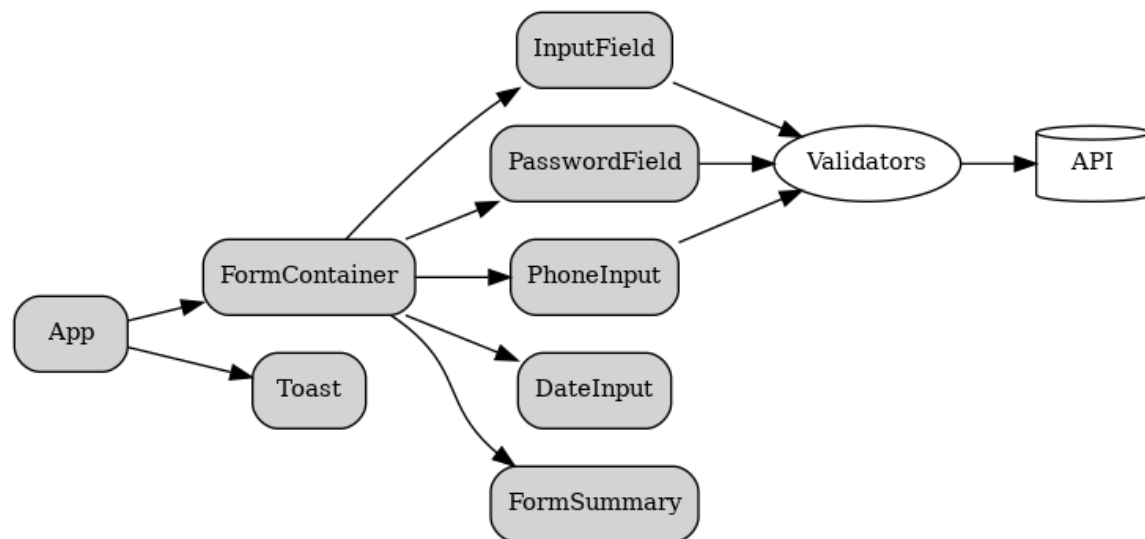
Suggested file / folder layout:

```
src/
├── components/
│   ├── FormContainer.jsx
│   ├── InputField.jsx
│   ├── PasswordField.jsx
│   ├── PhoneInput.jsx
│   ├── DateInput.jsx
```

```
│  ├─ FormSummary.jsx
│  └─ Toast.jsx
├─ validators/
│  ├─ index.js
│  ├─ email.js
│  ├─ password.js
│  ├─ username.js
│  └─ phone.js
├─ api/
│  └─ index.js
├─ hooks/
│  ├─ useDebounce.js
│  └─ useFormReducer.js
├─ utils/
│  └─ normalizers.js
└─ App.jsx
```
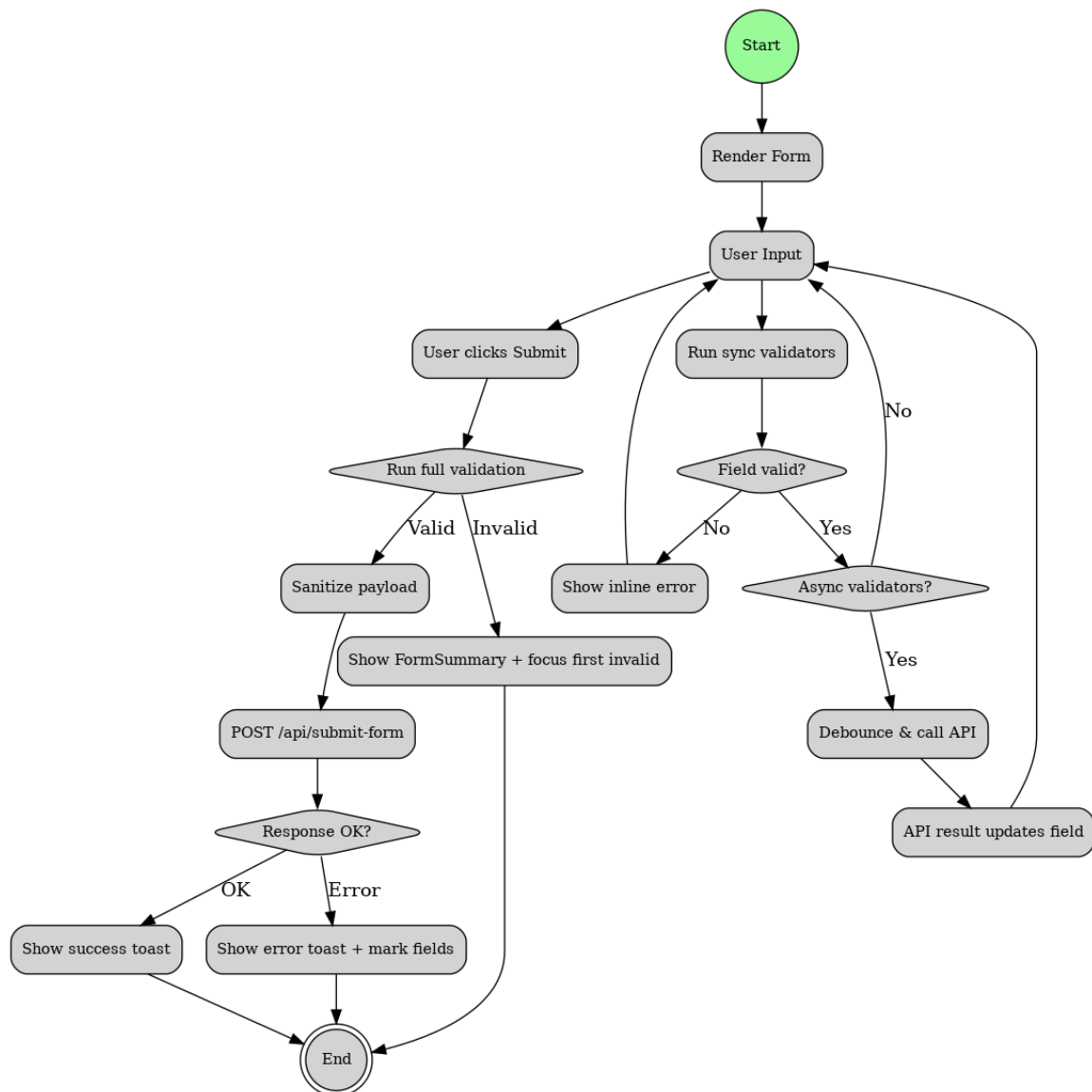
Component diagram (visual):



# Basic Flow Diagram

User interaction flow (high level):

• Render Registration page (FormContainer mounts).

• User types: synchronous validators run; async checks debounced.

• Field-level states update (valid/invalid, helper text, spinner).

• On submit: run full validation; if invalid show FormSummary and focus first invalid field.

• If valid: sanitize payload and POST to /api/submit-form; show success or error toast based on response.

Flow diagram (visual):



# Non-functional requirements & Deliverables

Non-functional requirements:

• Accessibility: ARIA attributes, live regions, keyboard-first navigation, color contrast (WCAG AA).

• Responsiveness: mobile-first design, modern browser support.

• Testability: unit tests for validators (target ≥ 80% coverage for core validation logic).

• Performance: debounce network checks, avoid heavy re-renders.

Deliverables for Phase 2 submission:

• Phase 2 — Solution Design & Architecture document (this file).

• Component and Flow diagrams (PNG/SVG).

• Sample API schema and payload examples.

• README with instructions to run demo and mock API.

• Optional: starter React component and validation utilities.

Prepared for: Skill Development Course — Frontend (Interactive Form Validation)

Author: Student