# Core Technical Documentation

## 1. Project Overview

Modern React-based permit application system built with **TypeScript**, featuring:

- Multi-step form wizard
- AI-powered text suggestions
- Accessibility-first design
- Robust testing infrastructure

---

## 2. Architecture Decisions

### 2.1 Core Component Library

**Stack**: Framer Motion + Tailwind CSS
**Key Components**:

- `FormInput` – animated input with validation & accessibility
- `FormSelect` – searchable dropdown, keyboard navigation, RTL
- `FormTextArea` – textarea with AI suggestions
- `ToastContainer` – animated notification system
- `RouteGuard` – step-based route protection

**Benefits**:

- Consistent animations

- WCAG 2.1 compliance
- RTL (Arabic) support
- TypeScript type safety

---

## 2.2 Multi-Step Form Architecture

- Route-based navigation (`/permit/personal`, `/permit/family-financial`, `/permit/situation`)
- State management via Redux
- Drafts persisted in LocalStorage
- Route protection with `RouteGuard`

**Benefits**:

- Code splitting & lazy loading
- Clear browser history management
- Easier debugging and testing

---

## 2.3 State Management

**Tool**: Redux Toolkit + RTK Query

- Centralized form state
- Predictable updates & time-travel debugging
- Efficient API calls with caching

---

## 2.4 Form Validation

**Tool**: Yup + React Hook Form

- Separate schemas per step
- Internationalized error messages
- Real-time validation

---

## 2.5 Accessibility

**WCAG 2.1 compliance**

- ARIA labels & roles
- Keyboard navigation (Tab, Arrows, Enter, Esc)
- Screen reader compatibility
- Focus management
- RTL + high-contrast support

---

## 2.6 AI Integration

**Tool**: OpenAI GPT API

- Field-specific, localized prompts (English/Arabic)
- Accept/Edit/Discard workflow
- Animated suggestion popup
- Error fallback handling

---

### 2.7 Data Persistence

- Drafts saved to LocalStorage
- Mock API for submissions
- Redux hydration from LocalStorage
- Validation before save

---

### 2.8 Internationalization

**Tool**: React i18next

- English & Arabic support
- Dynamic switching
- Localized validation messages
- RTL layout

---

### 2.9 Animations & Performance

**Tool**: Framer Motion

- Page transitions & micro-interactions
- Optimized animation variants
- Lazy loading & reduced motion support

---

### 2.10 Testing Strategy

- **Unit Tests**: Vitest + React Testing Library

- **E2E**: Playwright (with @axe-core accessibility)
- **Visual Regression**: Playwright screenshots
- **Coverage**: Vitest reporting

---

## 2.11 Development Workflow

- **Build**: Vite
- **Linting**: ESLint
- **Formatting**: Prettier + Tailwind plugin
- **Git Hooks**: Husky + lint-staged
- **CI/CD**: Strict type checking & coverage thresholds

---

## 2.12 Security

- Secure env vars for API keys
- Input sanitization
- XSS/CSRF prevention
- Safe localStorage usage

---

## 2.13 Performance Monitoring

- Route-level code splitting
- Lazy-loaded components
- Redux Toolkit efficiency

- Animation performance optimizations

---

### 2.14 Error Handling

- Global error boundaries
- Toast notifications
- Graceful API fallback
- Clear validation feedback

---

## 3. Technical Stack

- **Frontend**: React 19 + TypeScript + Vite
- **Styling**: Tailwind CSS 4.x
- **Animations**: Framer Motion
- **State Management**: Redux Toolkit + RTK Query
- **Forms**: React Hook Form + Yup
- **Routing**: React Router v7
- **i18n**: React i18next
- **Testing**: Vitest + Playwright
- **AI**: OpenAI GPT API
- **Accessibility**: WCAG 2.1 compliance

---

## 4. Key Achievements

1. Accessibility-first design (WCAG 2.1)
2. Route-based code splitting & performance optimization
3. AI-driven contextual text suggestions
4. Full RTL and multi-language support
5. Comprehensive unit, E2E & accessibility testing
6. Type-safe development with strict tooling
7. Smooth animations and robust error handling