System Architecture and Interface Overview (ACOP)

Phase: Project Design Phase **Date:** January 15, 2026 **Prepared By:** Development Team Lead
Document Status: Final v1.0 (Approved for Development)

1.0 Architectural Overview

The Automated Customer Onboarding Platform (ACOP) will utilize a modern **three-tier, cloud-native architecture** to ensure scalability, security, and high availability.

1.1 Technology Stack Summary

Layer	Component	Technology/Tool	Rationale
Presentation	Frontend User Interface	**React (SPA)**, Tailwind CSS	Highly responsive, component-based, fast rendering.
Application	Core Business Logic	**Python (Django REST Framework)**	Rapid development, strong security features, existing team expertise.
Data	Structured Data Store	**PostgreSQL**	Reliable relational data integrity for user and status tracking.
Storage	Unstructured Files	**Cloud Storage Bucket** (e.g., S3/GCS)	Scalable, secure, and cost-effective storage for documents (FR-3).

1.2 Component Structure

- **API Gateway:** Serves as the single entry point for all frontend requests, managing load balancing and applying authentication/authorization rules before routing to the backend services.
- **Customer Service:** Handles user registration, login, and profile management (FR-1).
- **Validation Service:** Executes the business logic for form validation and data quality checks (FR-2).
- **CRM Connector Service:** Dedicated service responsible for translating and sending finalized data to the CRM system (FR-4).

2.0 Key Interface Design Decisions (Wireframe Summary)

The design emphasizes clarity and simplicity to achieve NFR-3 (Usability).

- **Progress Indicator:** A mandatory, persistent 5-step progress bar will be displayed at the

top of every onboarding screen to prevent user abandonment by clearly indicating progress.

- **Key Screen:** The main data entry screen (Step 2) will use a **single-column layout** optimized for mobile view, ensuring all fields are easily tappable (NFR-3).
- **Document Upload:** A drag-and-drop zone will be implemented with clear size and format limits displayed to the user before upload, reducing friction.

3.0 Deployment and Environment Strategy

The system will be deployed using **Infrastructure-as-Code (IaC)** principles. Containers (Docker) will be used for all application services, ensuring consistency between development, testing, and production environments (NFR-1).