# KYC Architectural Design

## 1. Architectural Overview

The architecture can be broadly divided into the following components:  
1. Frontend/UI Layer: Built with modern frameworks such as Angular, React, or Vue.js.  
2. Backend Layer: Developed using Java Spring Boot, which includes RESTful APIs, business logic, and service orchestration.  
3. Database Layer: A relational database like PostgreSQL or MySQL for storing user information, KYC data, and audit logs.  
4. Integration Layer: Integration with external systems, such as identity verification services, anti-money laundering (AML) databases, and document storage solutions.  
5. Security Layer: Implemented with Spring Security for authentication and authorization, data encryption, and secure API communication.  
6. Workflow and Business Rules Engine: Use of a rules engine like Drools for handling dynamic business rules, and workflow orchestration with tools like Camunda or Activiti.

## 2. KYC Workflow Details

For each KYC workflow, we can define several stages such as registration, document upload, verification, risk assessment, and approval/rejection. Here is a breakdown of each stage:

### a. Registration Workflow

Frontend (UI Layer):  
- A user registration form is designed using UI components where users input their details (name, address, phone number, email, etc.).  
- Frontend validations are applied to ensure data quality (e.g., email format, mandatory fields).  
- Once the form is submitted, the data is sent to the backend via a REST API.  
  
Backend (Java Spring Boot Layer):  
- The backend receives user registration data and validates it using Java Bean Validation (JSR-303).  
- If valid, the user data is persisted in the relational database using JPA/Hibernate.  
- A confirmation email is sent to the user with a link to verify their email address (Spring Boot Mail Integration).

### b. Document Upload Workflow

Frontend (UI Layer):  
- A document upload interface where users upload their identity documents (passport, driver's license, utility bill, etc.).  
- Integrate a file upload component that supports formats like PDF, JPEG, and PNG.  
- Real-time feedback and progress bars show the upload status.  
  
Backend (Java Spring Boot Layer):  
- The backend API receives the uploaded documents and stores them in a secure document storage system (e.g., AWS S3, Azure Blob Storage).  
- Metadata related to the documents (e.g., file type, size, upload time) is stored in the database.  
- The documents are then processed using an OCR (Optical Character Recognition) service (e.g., Tesseract, Google Vision API) for text extraction and validation.

### c. Verification Workflow

Frontend (UI Layer):  
- A status dashboard shows users the progress of their KYC verification (e.g., 'In Progress', 'Verified', 'Rejected').  
- Notifications are sent to the user for any actions required (e.g., additional documents needed).  
  
Backend (Java Spring Boot Layer):  
- The verification process involves multiple services, such as Document Verification Service, Identity Verification Service, and AML Screening Service.  
- The results of the verification process are updated in the database, and relevant notifications are sent to the user.

### d. Risk Assessment Workflow

Backend (Java Spring Boot Layer):  
- A risk assessment service evaluates the user’s risk profile based on predefined rules and criteria (e.g., high-risk countries, politically exposed persons).  
- A rules engine (like Drools) can be used to dynamically assess risk factors based on changing regulatory requirements.  
- The risk score and assessment details are stored in the database for further review.

### e. Approval/Rejection Workflow

Frontend (UI Layer):  
- An admin dashboard is provided for compliance officers to review pending KYC applications.  
- Officers can approve or reject applications and provide feedback.  
  
Backend (Java Spring Boot Layer):  
- Based on the decision, the user’s KYC status is updated in the database, and the user is notified.  
- A complete audit trail of the approval/rejection process is maintained for compliance and reporting.

## 3. Technology Stack

Frontend: React.js or Angular, Redux for state management, Material-UI or Bootstrap for UI components.  
Backend: Spring Boot, Spring Data JPA, Spring Security, REST APIs.  
Database: PostgreSQL or MySQL for relational data, Elasticsearch for search and indexing.  
External Integrations: REST or SOAP-based APIs for identity verification and AML checks.  
Document Storage: AWS S3, Azure Blob Storage, or on-premises storage with encryption.  
Security: Spring Security, JWT (JSON Web Tokens) for authentication, HTTPS for secure communication.

## 4. Security Considerations

Implement multi-factor authentication (MFA) for user and admin logins.  
Use HTTPS and secure API communication with OAuth2.  
Store sensitive data encrypted both at rest (database, file storage) and in transit (API communication).  
Log and monitor all access and actions for audit trails and anomaly detection.