# Job Portal Technical Architecture

## Complete System Design Document for Unskilled Workers Platform in India

**Version:** 1.0  
**Date:** SEPTEMBER 2-25  
**Project:** Job Portal for Unskilled Workers  
**Target Market:** India  
**Document Type:** Complete Technical Design Document

## 📋 Table of Contents

1. [Executive Summary](https://claude.ai/chat/d1bd91a7-3a0d-492b-bffd-3cf5fad61294#1-executive-summary)
2. [System Architecture Overview](https://claude.ai/chat/d1bd91a7-3a0d-492b-bffd-3cf5fad61294#2-system-architecture-overview)
3. [Technology Stack](https://claude.ai/chat/d1bd91a7-3a0d-492b-bffd-3cf5fad61294#3-technology-stack)
4. [Database Design](https://claude.ai/chat/d1bd91a7-3a0d-492b-bffd-3cf5fad61294#4-database-design)
5. [API Architecture](https://claude.ai/chat/d1bd91a7-3a0d-492b-bffd-3cf5fad61294#5-api-architecture)
6. [Mobile Application Architecture](https://claude.ai/chat/d1bd91a7-3a0d-492b-bffd-3cf5fad61294#6-mobile-application-architecture)
7. [Backend Services](https://claude.ai/chat/d1bd91a7-3a0d-492b-bffd-3cf5fad61294#7-backend-services)
8. [Third-Party Integrations](https://claude.ai/chat/d1bd91a7-3a0d-492b-bffd-3cf5fad61294#8-third-party-integrations)
9. [Security Architecture](https://claude.ai/chat/d1bd91a7-3a0d-492b-bffd-3cf5fad61294#9-security-architecture)
10. [Scalability & Performance](https://claude.ai/chat/d1bd91a7-3a0d-492b-bffd-3cf5fad61294#10-scalability--performance)
11. [Deployment Architecture](https://claude.ai/chat/d1bd91a7-3a0d-492b-bffd-3cf5fad61294#11-deployment-architecture)
12. [Monitoring & Analytics](https://claude.ai/chat/d1bd91a7-3a0d-492b-bffd-3cf5fad61294#12-monitoring--analytics)

## 1. Executive Summary

### 1.1 Project Overview

The Job Portal is a comprehensive mobile-first platform designed specifically for unskilled workers in India. The system connects job seekers with employers while addressing unique challenges such as low digital literacy, limited smartphone capabilities, language barriers, and regional diversity.

### 1.2 Key Technical Requirements

* **Mobile-First Design**: Optimized for Android devices with 2-4GB RAM
* **Multi-Language Support**: Hindi, English, and other regional languages
* **Offline Capability**: Core functions work without internet connectivity
* **Low Bandwidth Optimization**: Minimal data usage for 2G/3G networks
* **Voice Integration**: Speech-to-text and text-to-speech capabilities
* **Scalability**: Support for 1M+ users across multiple cities
* **Security**: End-to-end encryption and compliance with Indian data protection laws

### 1.3 Target Performance Metrics

* **Response Time**: <2 seconds for core operations
* **Uptime**: 99.9% availability
* **Concurrent Users**: 50,000+ simultaneous users
* **Data Usage**: <10MB per user per month
* **Battery Optimization**: <5% battery drain per hour of usage
* **Load Capacity**: 10,000+ requests per second

## 2. System Architecture Overview

## 2.1 High-Level Architecture

─────────────────────────────────────────────────────────────────┐

│ CLIENT TIER │

├─────────────────┬─────────────────┬─────────────────────────────┤

│ Mobile Apps │ Web Portal │ Admin Panel │

│ (Android/iOS) │ (Employers) │ (Internal) │

│ │ │ │

│ - Job Seekers │ - Job Posting │ - User Management │

│ - Voice UI │ - Applications │ - Analytics │

│ - Offline Mode │ - Analytics │ - Content Moderation │

└─────────┬───────┴─────────┬───────┴─────────────┬───────────────┘

│ │ │

└─────────────────┼─────────────────────┘

│

┌─────────────────────────────────────────────────────────────────┐

│ API GATEWAY TIER │

├─────────────────────────────────────────────────────────────────┤

│ - Load Balancing - Rate Limiting │

│ - Authentication - Request Routing │

│ - API Versioning - Response Caching │

│ - SSL Termination - Request Logging │

└─────────────────────────┬───────────────────────────────────────┘

│

┌─────────────────────────────────────────────────────────────────┐

│ MICROSERVICES TIER │

├──────────────┬──────────────┬──────────────┬───────────────────┤

│ Auth Service │ Job Service │ User Service │ Notification Svc │

│ │ │ │ │

│- OTP Login │- Job CRUD │- Profiles │- Push/SMS │

│- JWT Tokens │- Search │- Skills │- Email │

│- Sessions │- Matching │- Documents │- In-App │

│- Refresh │- Analytics │- Verification│- Voice Messages │

├──────────────┼──────────────┼──────────────┼───────────────────┤

│Payment Svc │Message Svc │Location Svc │ File Service │

│ │ │ │ │

│- Razorpay │- Chat │- GPS/Maps │- Upload/Download │

│- UPI │- Voice Msgs │- Geocoding │- Image Processing │

│- Escrow │- Push Notif │- Distance │- Document OCR │

│- Commission │- SMS Backup │- Radius │- CDN Integration │

└──────────────┴──────────────┴──────────────┴───────────────────┘

│

┌─────────────────────────────────────────────────────────────────┐

│ DATA TIER │

├─────────────────┬─────────────────┬─────────────────────────────┤

│ PostgreSQL │ MongoDB │ Redis │

│ │ │ │

│- User Data │- Documents │- Sessions │

│- Job Posts │- Analytics │- Cache │

│- Applications │- Logs │- Queues │

│- Transactions │- Files Metadata │- Real-time Data │

├─────────────────┼─────────────────┼─────────────────────────────┤

│ Elasticsearch │ File Storage │ Message Queue │

│ │ │ │

│- Job Search │- AWS S3 │- Bull Queue │

│- User Search │- Images │- Job Processing │

│- Analytics │- Documents │- Email Queue │

│- Logging │- Voice Files │- Notification Queue │

└─────────────────┴─────────────────┴─────────────────────────────┘

### 2.2 Architecture Principles

#### 2.2.1 Microservices Architecture

* **Service Independence**: Each service can be developed, deployed, and scaled independently
* **Domain-Driven Design**: Services organized around business capabilities
* **Fault Isolation**: Failure in one service doesn't bring down the entire system
* **Technology Diversity**: Different services can use different technologies as needed
* **Team Autonomy**: Small teams can own entire service lifecycles

#### 2.2.2 API-First Design

* **Contract-First Development**: APIs defined before implementation
* **Comprehensive Documentation**: Swagger/OpenAPI specifications for all endpoints
* **Versioning Strategy**: Backward compatibility with version management
* **Consistent Standards**: Uniform naming conventions and response formats
* **Rate Limiting**: Protection against abuse and ensuring fair usage

#### 2.2.3 Mobile-First Approach

* **Progressive Web App (PWA)**: Web app with native-like capabilities
* **Offline-First**: Core functionality available without internet
* **Performance Optimization**: Minimal bundle sizes and lazy loading
* **Device Compatibility**: Support for low-end Android devices
* **Battery Efficiency**: Optimized background processes

#### 2.2.4 Data-Driven Architecture

* **Real-time Analytics**: Immediate insights into user behavior
* **Machine Learning Integration**: AI-powered job matching and recommendations
* **Performance Monitoring**: Continuous optimization based on usage patterns
* **Business Intelligence**: Comprehensive reporting and dashboards
* **A/B Testing**: Data-driven feature development and optimization

## 3. Technology Stack

### 3.1 Frontend Technologies

#### 3.1.1 Mobile Application (React Native)

Core Framework:

- React Native: 0.72+

- TypeScript: 5.0+

- Metro Bundler: Latest

State Management:

- Redux Toolkit: 1.9+

- RTK Query: Data fetching and caching

- React Redux: 8.0+

- Redux Persist: State persistence

Navigation:

- React Navigation: 6.x

- Stack Navigator: Screen transitions

- Tab Navigator: Bottom tabs

- Drawer Navigator: Side menu

UI Components:

- NativeBase: 3.4+

- React Native Elements: Fallback

- React Native Vector Icons: 10+

- React Native Gesture Handler: 2.12+

Device Features:

- React Native Maps: Location services

- React Native Voice: Speech recognition

- React Native TTS: Text-to-speech

- React Native Camera: Photo capture

- React Native Audio: Voice recording

- React Native Geolocation: GPS access

Offline & Storage:

- SQLite: Local database

- WatermelonDB: Reactive local database

- React Native Async Storage: Key-value storage

- MMKV: Fast key-value storage

- React Native FS: File system access

Push Notifications:

- Firebase Cloud Messaging: Cross-platform

- React Native Notifications: Local notifications

- React Native Push Notification: iOS support

Development Tools:

- Flipper: Debugging

- Reactotron: Development tool

- CodePush: Over-the-air updates

- Sentry: Error tracking

- Jest: Unit testing

- Detox: E2E testing

#### 3.1.2 Web Portal (Employer Dashboard)

Core Framework:

- Next.js: 13+ with App Router

- TypeScript: 5.0+

- React: 18+

Styling:

- Tailwind CSS: 3.3+

- Headless UI: Unstyled components

- Heroicons: Icon library

- Framer Motion: Animations

State Management:

- Zustand: Lightweight state management

- TanStack Query: Server state management

- SWR: Data fetching (alternative)

Authentication:

- NextAuth.js: Authentication library

- JWT: Token handling

- OAuth providers: Google, LinkedIn

Data Visualization:

- Chart.js: Charts and graphs

- D3.js: Custom visualizations

- Recharts: React charts

Forms & Validation:

- React Hook Form: Form handling

- Zod: Schema validation

- Yup: Alternative validation

Development Tools:

- ESLint: Code linting

- Prettier: Code formatting

- Jest: Unit testing

- Cypress: E2E testing

- Storybook: Component documentation

#### 3.1.3 Admin Dashboard

Core Framework:

- React: 18+

- TypeScript: 5.0+

- Vite: Build tool

UI Library:

- Ant Design: 5.0+

- Material-UI: Alternative

- Styled Components: CSS-in-JS

State Management:

- Redux Toolkit: Complex state

- React Query: Server state

Data Visualization:

- D3.js: Advanced visualizations

- Recharts: React charts

- Victory: Alternative charts

- MapBox: Geographic data

Advanced Features:

- React Virtual: Large lists

- React DnD: Drag and drop

- React Helmet: SEO

- React Router: Client-side routing

### 3.2 Backend Technologies

#### 3.2.1 Core API Services

Runtime Environment:

- Node.js: 18 LTS

- npm: 9+

- TypeScript: 5.0+

Web Framework:

- Express.js: 4.18+

- Helmet: Security middleware

- CORS: Cross-origin requests

- Compression: Response compression

Authentication:

- Passport.js: Authentication middleware

- JWT: JSON Web Tokens

- bcrypt: Password hashing

- speakeasy: Two-factor authentication

API Documentation:

- Swagger: API documentation

- OpenAPI: 3.0 specification

- Redoc: Alternative documentation

Validation & Sanitization:

- Joi: Schema validation

- express-validator: Request validation

- DOMPurify: XSS protection

- Helmet: Security headers

File Handling:

- Multer: File upload middleware

- Sharp: Image processing

- pdf-parse: PDF text extraction

- archiver: File compression

Background Jobs:

- Bull Queue: Job processing

- Redis: Queue backend

- Cron: Scheduled tasks

- node-schedule: Job scheduling

Testing:

- Jest: Testing framework

- Supertest: API testing

- MongoDB Memory Server: Test database

- Factory Girl: Test data generation

#### 3.2.2 Database Systems

Primary Database:

- PostgreSQL: 15+

- PostGIS: Geographic data

- pg: Node.js driver

- Sequelize: ORM (alternative)

- TypeORM: TypeScript ORM

Document Database:

- MongoDB: 6.x

- Mongoose: ODM

- MongoDB Compass: GUI tool

Cache & Session Store:

- Redis: 7.x

- ioredis: Node.js client

- Redis Sentinel: High availability

- Redis Cluster: Horizontal scaling

Search Engine:

- Elasticsearch: 8.x

- Kibana: Visualization

- Logstash: Data pipeline

- APM: Application monitoring

Time Series Database:

- InfluxDB: Metrics storage

- Grafana: Visualization

- Telegraf: Data collection

#### 3.2.3 Additional Backend Services

Email Services:

- SendGrid: Transactional emails

- Amazon SES: Alternative

- Nodemailer: SMTP client

- Email Templates: Handlebars

SMS Services:

- Twilio: International SMS

- MSG91: India-focused SMS

- AWS SNS: Alternative

- Bulk SMS: Mass messaging

Push Notifications:

- Firebase Admin SDK: FCM server

- Apple Push Notifications: iOS

- Web Push: Browser notifications

Payment Processing:

- Razorpay: Indian payment gateway

- Stripe: International payments

- PayPal: Alternative

- UPI: Direct bank transfers

File Storage:

- AWS S3: Object storage

- Google Cloud Storage: Alternative

- Cloudinary: Image optimization

- ImageKit: Image CDN

PDF Generation:

- Puppeteer: PDF from HTML

- jsPDF: Client-side PDF

- PDFKit: Server-side PDF

Media Processing:

- FFmpeg: Video/audio processing

- ImageMagick: Image manipulation

- GraphicsMagick: Alternative

### 3.3 DevOps & Infrastructure

#### 3.3.1 Containerization & Orchestration

Containerization:

- Docker: 24+

- Docker Compose: Multi-container apps

- Dockerfile: Container definitions

- .dockerignore: Optimization

Container Orchestration:

- Kubernetes: 1.28+

- Helm: Package manager

- Ingress Controller: Traffic routing

- Cert Manager: SSL certificates

Container Registry:

- Docker Hub: Public images

- AWS ECR: Private registry

- GitHub Container Registry: CI/CD integration

#### 3.3.2 CI/CD Pipeline

Version Control:

- Git: Source control

- GitHub: Repository hosting

- GitLab: Alternative platform

- Bitbucket: Atlassian integration

CI/CD Platforms:

- GitHub Actions: Integrated CI/CD

- GitLab CI: Alternative

- Jenkins: Self-hosted

- CircleCI: Cloud-based

Continuous Integration:

- Automated testing: Unit, integration, e2e

- Code quality: ESLint, SonarQube

- Security scanning: Snyk, OWASP

- Dependency checking: npm audit

Continuous Deployment:

- Blue-green deployment: Zero downtime

- Rolling updates: Gradual rollout

- Canary releases: Risk mitigation

- Feature flags: Controlled rollout

#### 3.3.3 Infrastructure as Code

Infrastructure Provisioning:

- Terraform: 1.5+

- AWS CloudFormation: AWS native

- Pulumi: Multi-cloud

- Ansible: Configuration management

Configuration Management:

- Kubernetes ConfigMaps: Application config

- Kubernetes Secrets: Sensitive data

- Vault: Secret management

- AWS Systems Manager: Parameter store

Environment Management:

- Development: Local and cloud

- Staging: Production-like testing

- Production: Live environment

- DR: Disaster recovery

#### 3.3.4 Monitoring & Observability

Application Monitoring:

- Prometheus: Metrics collection

- Grafana: Visualization

- AlertManager: Alert routing

- Jaeger: Distributed tracing

Log Management:

- ELK Stack: Elasticsearch, Logstash, Kibana

- Fluentd: Log forwarding

- CloudWatch: AWS logging

- Papertrail: Log aggregation

Error Tracking:

- Sentry: Error monitoring

- Bugsnag: Alternative

- Rollbar: Error reporting

- LogRocket: Session replay

Performance Monitoring:

- New Relic: APM

- DataDog: Infrastructure monitoring

- Dynatrace: Full-stack monitoring

- Pingdom: Uptime monitoring

Security Monitoring:

- AWS GuardDuty: Threat detection

- Falco: Runtime security

- OWASP ZAP: Security testing

- Nessus: Vulnerability scanning

### 3.4 Third-Party Services

#### 3.4.1 Communication Services

SMS Gateways:

- Twilio: Global SMS service

- MSG91: India-focused SMS

- TextLocal: UK-based service

- AWS SNS: Amazon's SMS service

Email Services:

- SendGrid: Transactional emails

- Mailgun: Developer-friendly

- Amazon SES: AWS email service

- Postmark: High deliverability

Voice Services:

- Twilio Voice: Voice calls

- Nexmo/Vonage: Voice API

- Amazon Connect: Contact center

- Google Cloud Speech: Speech recognition

#### 3.4.2 Payment & Financial Services

Payment Gateways:

- Razorpay: Indian payment solutions

- Stripe: Global payments

- PayPal: International payments

- Paytm: Indian digital wallet

UPI Providers:

- PhonePe: UPI transactions

- Google Pay: Google's UPI

- Paytm UPI: Wallet integration

- BHIM: Government UPI app

Banking APIs:

- Razorpay X: Banking solutions

- Cashfree: Payment solutions

- InstaMojo: Small business payments

- CCAvenue: Indian payment gateway

#### 3.4.3 Location & Maps Services

Mapping Services:

- Google Maps: Comprehensive mapping

- MapBox: Customizable maps

- Here Maps: Enterprise solutions

- OpenStreetMap: Open-source maps

Geocoding Services:

- Google Geocoding API: Address conversion

- MapBox Geocoding: Alternative

- Nominatim: OSM geocoding

- Here Geocoding: Enterprise

Location Intelligence:

- Google Places API: POI data

- Foursquare Places: Location data

- Yelp API: Business information

- TomTom Places: Location services

#### 3.4.4 Identity & Document Verification

Indian Identity Verification:

- Aadhaar API: Government ID verification

- PAN Verification: Tax ID verification

- Driving License API: DL verification

- Voter ID API: Electoral ID verification

Document Processing:

- Tesseract OCR: Text extraction

- Google Vision API: Document analysis

- Amazon Textract: Document processing

- Azure Cognitive Services: AI services

KYC Services:

- IDfy: Identity verification

- Signzy: Digital KYC

- Perfios: Financial verification

- Bureau ID: Identity solutions

## 4. Database Design

### 4.1 PostgreSQL Schema Design

#### 4.1.1 Core User Management Tables

-- Enable necessary extensions

CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

CREATE EXTENSION IF NOT EXISTS "postgis";

CREATE EXTENSION IF NOT EXISTS "pg\_trgm";

CREATE EXTENSION IF NOT EXISTS "btree\_gin";

-- Custom types for better data integrity

CREATE TYPE user\_type\_enum AS ENUM ('job\_seeker', 'employer', 'admin', 'support');

CREATE TYPE user\_status\_enum AS ENUM ('active', 'suspended', 'pending', 'deactivated');

CREATE TYPE gender\_enum AS ENUM ('male', 'female', 'other', 'prefer\_not\_to\_say');

CREATE TYPE business\_type\_enum AS ENUM ('individual', 'small\_business', 'company', 'startup', 'ngo');

CREATE TYPE verification\_status\_enum AS ENUM ('pending', 'in\_progress', 'verified', 'rejected', 'expired');

CREATE TYPE experience\_level\_enum AS ENUM ('fresher', 'beginner', 'intermediate', 'experienced', 'expert');

CREATE TYPE availability\_enum AS ENUM ('available', 'busy', 'not\_available', 'partially\_available');

-- Users table (both job seekers and employers)

CREATE TABLE users (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

phone\_number VARCHAR(15) UNIQUE NOT NULL,

country\_code VARCHAR(5) DEFAULT '+91',

email VARCHAR(255) UNIQUE,

email\_verified BOOLEAN DEFAULT FALSE,

user\_type user\_type\_enum NOT NULL,

status user\_status\_enum DEFAULT 'pending',

preferred\_language VARCHAR(10) DEFAULT 'hi',

timezone VARCHAR(50) DEFAULT 'Asia/Kolkata',

last\_login\_at TIMESTAMP,

login\_count INTEGER DEFAULT 0,

terms\_accepted\_at TIMESTAMP,

privacy\_accepted\_at TIMESTAMP,

marketing\_consent BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW(),

deleted\_at TIMESTAMP,

-- Constraints

CONSTRAINT users\_phone\_format CHECK (phone\_number ~ '^[0-9]{10,15}$'),

CONSTRAINT users\_email\_format CHECK (email ~ '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$')

);

-- Create indexes for performance

CREATE INDEX CONCURRENTLY idx\_users\_phone ON users(phone\_number) WHERE deleted\_at IS NULL;

CREATE INDEX CONCURRENTLY idx\_users\_email ON users(email) WHERE deleted\_at IS NULL;

CREATE INDEX CONCURRENTLY idx\_users\_type\_status ON users(user\_type, status) WHERE deleted\_at IS NULL;

CREATE INDEX CONCURRENTLY idx\_users\_created\_at ON users(created\_at DESC);

-- Job Seeker Profiles

CREATE TABLE job\_seeker\_profiles (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

user\_id UUID REFERENCES users(id) ON DELETE CASCADE,

full\_name VARCHAR(255) NOT NULL,

display\_name VARCHAR(100),

date\_of\_birth DATE,

age INTEGER GENERATED ALWAYS AS (EXTRACT(YEAR FROM AGE(date\_of\_birth))) STORED,

gender gender\_enum,

profile\_photo\_url VARCHAR(500),

cover\_image\_url VARCHAR(500),

-- Contact Information

whatsapp\_number VARCHAR(15),

emergency\_contact\_name VARCHAR(255),

emergency\_contact\_phone VARCHAR(15),

-- Address Information

current\_location POINT, -- PostGIS geography type

address\_line1 VARCHAR(255),

address\_line2 VARCHAR(255),

landmark VARCHAR(255),

city VARCHAR(100) NOT NULL,

district VARCHAR(100),

state VARCHAR(100) NOT NULL,

country VARCHAR(100) DEFAULT 'India',

pincode VARCHAR(10),

-- Work Preferences

preferred\_work\_radius INTEGER DEFAULT 10, -- in kilometers

max\_commute\_time INTEGER DEFAULT 60, -- in minutes

availability availability\_enum DEFAULT 'available',

available\_days JSONB DEFAULT '["monday","tuesday","wednesday","thursday","friday","saturday"]',

preferred\_shift JSONB DEFAULT '{"morning": true, "afternoon": true, "evening": false, "night": false}',

willing\_to\_relocate BOOLEAN DEFAULT FALSE,

-- Financial Information

expected\_salary\_min INTEGER,

expected\_salary\_max INTEGER,

current\_salary INTEGER,

salary\_currency VARCHAR(3) DEFAULT 'INR',

salary\_type VARCHAR(20) DEFAULT 'monthly', -- hourly, daily, weekly, monthly

-- Personal Information

languages\_known JSONB DEFAULT '[]', -- Array of language codes

education\_level VARCHAR(50),

marital\_status VARCHAR(20),

family\_members INTEGER,

dependents INTEGER,

-- Verification Status

verification\_status JSONB DEFAULT '{"identity": "pending", "address": "pending", "phone": "pending"}',

verification\_score INTEGER DEFAULT 0, -- 0-100

trust\_score DECIMAL(3,2) DEFAULT 0.0, -- 0.00-5.00

-- Profile Completion

profile\_completion\_percentage INTEGER DEFAULT 0,

onboarding\_completed BOOLEAN DEFAULT FALSE,

onboarding\_step VARCHAR(50) DEFAULT 'basic\_info',

-- Activity Tracking

last\_active\_at TIMESTAMP DEFAULT NOW(),

job\_search\_count INTEGER DEFAULT 0,

applications\_sent INTEGER DEFAULT 0,

profile\_views INTEGER DEFAULT 0,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW(),

-- Constraints

CONSTRAINT age\_valid CHECK (age >= 16 AND age <= 80),

CONSTRAINT salary\_valid CHECK (expected\_salary\_min <= expected\_salary\_max),

CONSTRAINT radius\_valid CHECK (preferred\_work\_radius >= 1 AND preferred\_work\_radius <= 100)

);

-- Indexes for job seeker profiles

CREATE INDEX CONCURRENTLY idx\_job\_seeker\_user\_id ON job\_seeker\_profiles(user\_id);

CREATE INDEX CONCURRENTLY idx\_job\_seeker\_location ON job\_seeker\_profiles USING GIST(current\_location);

CREATE INDEX CONCURRENTLY idx\_job\_seeker\_city\_state ON job\_seeker\_profiles(city, state);

CREATE INDEX CONCURRENTLY idx\_job\_seeker\_availability ON job\_seeker\_profiles(availability) WHERE availability != 'not\_available';

CREATE INDEX CONCURRENTLY idx\_job\_seeker\_salary ON job\_seeker\_profiles(expected\_salary\_min, expected\_salary\_max);

CREATE INDEX CONCURRENTLY idx\_job\_seeker\_active ON job\_seeker\_profiles(last\_active\_at DESC);

-- Employer Profiles

CREATE TABLE employer\_profiles (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

user\_id UUID REFERENCES users(id) ON DELETE CASCADE,

-- Business Information

company\_name VARCHAR(255) NOT NULL,

display\_name VARCHAR(100),

company\_description TEXT,

website\_url VARCHAR(500),

linkedin\_url VARCHAR(500),

contact\_person\_name VARCHAR(255) NOT NULL,

designation VARCHAR(100),

-- Business Details

business\_type business\_type\_enum NOT NULL,

industry\_category VARCHAR(100),

company\_size VARCHAR(50), -- '1-10', '11-50', '51-200', '201-500', '500+'

founded\_year INTEGER,

-- Registration Details

business\_registration\_number VARCHAR(100),

gst\_number VARCHAR(15),

pan\_number VARCHAR(10),

tan\_number VARCHAR(10),

-- Contact Information

business\_phone VARCHAR(15),

business\_email VARCHAR(255),

hr\_contact\_name VARCHAR(255),

hr\_contact\_phone VARCHAR(15),

hr\_contact\_email VARCHAR(255),

-- Address Information

business\_address JSONB NOT NULL, -- {"line1": "", "line2": "", "city": "", "state": "", "pincode": ""}

headquarters\_location POINT,

branch\_locations JSONB DEFAULT '[]',

service\_areas JSONB DEFAULT '[]', -- Areas where they hire

-- Verification Information

verification\_documents JSONB DEFAULT '[]', -- Array of document references

verification\_status verification\_status\_enum DEFAULT 'pending',

verification\_date TIMESTAMP,

verified\_by UUID REFERENCES users(id),

-- Business Metrics

rating DECIMAL(3,2) DEFAULT 0.0,

total\_reviews INTEGER DEFAULT 0,

total\_jobs\_posted INTEGER DEFAULT 0,

total\_hires INTEGER DEFAULT 0,

successful\_hires INTEGER DEFAULT 0,

average\_response\_time INTEGER, -- in hours

-- Profile Information

company\_logo\_url VARCHAR(500),

cover\_image\_url VARCHAR(500),

gallery\_images JSONB DEFAULT '[]',

-- Subscription & Premium Features

subscription\_type VARCHAR(50) DEFAULT 'free', -- free, basic, premium, enterprise

subscription\_expires\_at TIMESTAMP,

monthly\_job\_limit INTEGER DEFAULT 5,

jobs\_posted\_this\_month INTEGER DEFAULT 0,

-- Activity Tracking

last\_active\_at TIMESTAMP DEFAULT NOW(),

last\_job\_posted\_at TIMESTAMP,

profile\_views INTEGER DEFAULT 0,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW(),

-- Constraints

CONSTRAINT gst\_format CHECK (gst\_number ~ '^[0-9]{2}[A-Z]{5}[0-9]{4}[A-Z]{1}[1-9A-Z]{1}Z[0-9A-Z]{1}$'),

CONSTRAINT pan\_format CHECK (pan\_number ~ '^[A-Z]{5}[0-9]{4}[A-Z]{1}$'),

CONSTRAINT rating\_valid CHECK (rating >= 0.0 AND rating <= 5.0)

);

-- Indexes for employer profiles

CREATE INDEX CONCURRENTLY idx\_employer\_user\_id ON employer\_profiles(user\_id);

CREATE INDEX CONCURRENTLY idx\_employer\_company\_name ON employer\_profiles USING GIN(company\_name gin\_trgm\_ops);

CREATE INDEX CONCURRENTLY idx\_employer\_industry ON employer\_profiles(industry\_category);

CREATE INDEX CONCURRENTLY idx\_employer\_verification ON employer\_profiles(verification\_status);

CREATE INDEX CONCURRENTLY idx\_employer\_location ON employer\_profiles USING GIST(headquarters\_location);

CREATE INDEX CONCURRENTLY idx\_employer\_rating ON employer\_profiles(rating DESC);

CREATE INDEX CONCURRENTLY idx\_employer\_active ON employer\_profiles(last\_active\_at DESC);

**4.1.2 Skills and Categories System**

-- Skill Categories (Construction, Delivery, etc.)

CREATE TABLE skill\_categories (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name VARCHAR(100) NOT NULL,

name\_hindi VARCHAR(100),

name\_regional JSONB DEFAULT '{}', -- {"te": "Telugu name", "ta": "Tamil name", ...}

slug VARCHAR(100) UNIQUE NOT NULL,

description TEXT,

icon\_url VARCHAR(500),

color\_code VARCHAR(7) DEFAULT '#6B7280', -- Hex color for UI

-- Hierarchy Support

parent\_category\_id UUID REFERENCES skill\_categories(id),

level INTEGER DEFAULT 1, -- 1 = main category, 2 = subcategory

sort\_order INTEGER DEFAULT 0,

-- Requirements

requires\_verification BOOLEAN DEFAULT FALSE,

min\_age INTEGER DEFAULT 16,

max\_age INTEGER,

physical\_requirements JSONB DEFAULT '[]',

-- Metadata

is\_active BOOLEAN DEFAULT TRUE,

is\_featured BOOLEAN DEFAULT FALSE,

job\_count INTEGER DEFAULT 0, -- Updated via triggers

worker\_count INTEGER DEFAULT 0,

avg\_salary DECIMAL(10,2),

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW(),

created\_by UUID REFERENCES users(id)

);

-- Individual Skills under categories

CREATE TABLE skills (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

category\_id UUID REFERENCES skill\_categories(id) NOT NULL,

name VARCHAR(100) NOT NULL,

name\_hindi VARCHAR(100),

name\_regional JSONB DEFAULT '{}',

slug VARCHAR(100) UNIQUE NOT NULL,

description TEXT,

-- Skill Properties

difficulty\_level experience\_level\_enum DEFAULT 'beginner',

learning\_time\_days INTEGER, -- Estimated days to learn

requires\_certification BOOLEAN DEFAULT FALSE,

requires\_verification BOOLEAN DEFAULT FALSE,

verification\_type VARCHAR(50) DEFAULT 'self\_declared', -- self\_declared, online\_test, practical, document

-- Related Skills

prerequisite\_skills JSONB DEFAULT '[]', -- Array of skill IDs

related\_skills JSONB DEFAULT '[]',

alternative\_names JSONB DEFAULT '[]',

-- Market Data

demand\_score INTEGER DEFAULT 0, -- 0-100 based on job postings

supply\_score INTEGER DEFAULT 0, -- 0-100 based on workers

avg\_salary\_min INTEGER,

avg\_salary\_max INTEGER,

job\_count INTEGER DEFAULT 0,

worker\_count INTEGER DEFAULT 0,

-- Content

training\_resources JSONB DEFAULT '[]',

video\_url VARCHAR(500),

documentation\_url VARCHAR(500),

is\_active BOOLEAN DEFAULT TRUE,

is\_trending BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW(),

created\_by UUID REFERENCES users(id)

);

-- Job Seeker Skills mapping with proficiency and verification

CREATE TABLE job\_seeker\_skills (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

job\_seeker\_id UUID REFERENCES job\_seeker\_profiles(id) ON DELETE CASCADE,

skill\_id UUID REFERENCES skills(id) ON DELETE CASCADE,

-- Skill Level

experience\_level experience\_level\_enum NOT NULL,

years\_of\_experience INTEGER DEFAULT 0,

months\_of\_experience INTEGER DEFAULT 0,

self\_rating INTEGER DEFAULT 1, -- 1-10 scale

-- Verification

verification\_status VARCHAR(50) DEFAULT 'unverified', -- unverified, in\_progress, verified, rejected

verification\_date TIMESTAMP,

verification\_score INTEGER, -- 0-100

verification\_method VARCHAR(50), -- test, practical, document, reference

verification\_details JSONB DEFAULT '{}',

verified\_by UUID REFERENCES users(id),

-- Experience Details

work\_samples JSONB DEFAULT '[]', -- URLs to work samples

certifications JSONB DEFAULT '[]', -- Certificate details

training\_completed JSONB DEFAULT '[]', -- Training records

projects\_completed INTEGER DEFAULT 0,

-- Usage Tracking

skill\_usage\_frequency VARCHAR(20) DEFAULT 'occasionally', -- daily, weekly, monthly, occasionally, rarely

last\_used\_at TIMESTAMP,

willing\_to\_teach BOOLEAN DEFAULT FALSE,

available\_for\_training BOOLEAN DEFAULT TRUE,

-- Metadata

added\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW(),

is\_primary\_skill BOOLEAN DEFAULT FALSE,

is\_featured BOOLEAN DEFAULT FALSE,

UNIQUE(job\_seeker\_id, skill\_id)

);

-- Indexes for skills system

CREATE INDEX CONCURRENTLY idx\_skill\_categories\_slug ON skill\_categories(slug);

CREATE INDEX CONCURRENTLY idx\_skill\_categories\_parent ON skill\_categories(parent\_category\_id);

CREATE INDEX CONCURRENTLY idx\_skills\_category ON skills(category\_id);

CREATE INDEX CONCURRENTLY idx\_skills\_slug ON skills(slug);

CREATE INDEX CONCURRENTLY idx\_job\_seeker\_skills\_seeker ON job\_seeker\_skills(job\_seeker\_id);

CREATE INDEX CONCURRENTLY idx\_job\_seeker\_skills\_skill ON job\_seeker\_skills(skill\_id);

CREATE INDEX CONCURRENTLY idx\_job\_seeker\_skills\_verification ON job\_seeker\_skills(verification\_status);

**4.1.3 Job Postings and Applications System**

-- Create enums for job-related data

CREATE TYPE payment\_type\_enum AS ENUM ('hourly', 'daily', 'weekly', 'monthly', 'fixed', 'negotiable');

CREATE TYPE urgency\_enum AS ENUM ('immediate', 'within\_24\_hours', 'within\_week', 'within\_month', 'flexible');

CREATE TYPE duration\_type\_enum AS ENUM ('hours', 'days', 'weeks', 'months', 'permanent', 'contract');

CREATE TYPE job\_status\_enum AS ENUM ('draft', 'active', 'paused', 'filled', 'expired', 'cancelled');

CREATE TYPE application\_status\_enum AS ENUM ('applied', 'viewed', 'shortlisted', 'interview\_scheduled', 'interviewed', 'selected', 'rejected', 'withdrawn', 'expired');

CREATE TYPE interview\_type\_enum AS ENUM ('phone', 'video', 'in\_person', 'practical', 'written');

-- Job Postings

CREATE TABLE job\_postings (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

employer\_id UUID REFERENCES employer\_profiles(id) ON DELETE CASCADE,

-- Basic Job Information

title VARCHAR(255) NOT NULL,

description TEXT NOT NULL,

short\_description VARCHAR(500),

category\_id UUID REFERENCES skill\_categories(id),

job\_type VARCHAR(50) DEFAULT 'full\_time', -- full\_time, part\_time, contract, temporary, internship

-- Skills Requirements

required\_skills JSONB NOT NULL, -- [{"skill\_id": "uuid", "required": true, "min\_experience": 2}, ...]

optional\_skills JSONB DEFAULT '[]',

skill\_requirements\_flexible BOOLEAN DEFAULT TRUE,

-- Location Information

job\_location POINT NOT NULL, -- PostGIS geography

location\_address JSONB NOT NULL, -- Structured address

location\_description VARCHAR(500),

search\_radius INTEGER DEFAULT 5, -- in kilometers

remote\_work\_allowed BOOLEAN DEFAULT FALSE,

on\_site\_required BOOLEAN DEFAULT TRUE,

travel\_required BOOLEAN DEFAULT FALSE,

-- Compensation Details

payment\_type payment\_type\_enum NOT NULL,

salary\_min DECIMAL(10,2),

salary\_max DECIMAL(10,2),

salary\_currency VARCHAR(3) DEFAULT 'INR',

salary\_negotiable BOOLEAN DEFAULT FALSE,

-- Additional Benefits

benefits JSONB DEFAULT '[]', -- ["food\_provided", "transport\_allowance", "medical\_insurance"]

bonus\_details TEXT,

overtime\_pay BOOLEAN DEFAULT FALSE,

performance\_incentives BOOLEAN DEFAULT FALSE,

-- Job Requirements

experience\_required experience\_level\_enum,

min\_age INTEGER DEFAULT 16,

max\_age INTEGER,

gender\_preference gender\_enum,

education\_requirements VARCHAR(500),

physical\_requirements TEXT,

additional\_requirements TEXT,

language\_requirements JSONB DEFAULT '[]',

-- Timeline and Schedule

urgency urgency\_enum NOT NULL,

start\_date DATE,

end\_date DATE,

duration\_type duration\_type\_enum,

duration\_value INTEGER,

working\_hours JSONB DEFAULT '{}', -- {"start": "09:00", "end": "18:00", "break": "60"}

working\_days JSONB DEFAULT '[]', -- ["monday", "tuesday", ...]

shift\_details TEXT,

-- Application Process

application\_deadline TIMESTAMP,

application\_instructions TEXT,

contact\_person VARCHAR(255),

contact\_phone VARCHAR(15),

contact\_email VARCHAR(255),

application\_url VARCHAR(500),

-- Job Posting Status

status job\_status\_enum DEFAULT 'draft',

is\_featured BOOLEAN DEFAULT FALSE,

is\_premium BOOLEAN DEFAULT FALSE,

boost\_level INTEGER DEFAULT 0, -- 0=normal, 1-5=boosted visibility

-- Metrics and Tracking

applications\_count INTEGER DEFAULT 0,

views\_count INTEGER DEFAULT 0,

unique\_views\_count INTEGER DEFAULT 0,

shortlist\_count INTEGER DEFAULT 0,

interview\_count INTEGER DEFAULT 0,

hire\_count INTEGER DEFAULT 0,

-- Auto-generated fields

expires\_at TIMESTAMP,

published\_at TIMESTAMP,

filled\_at TIMESTAMP,

last\_activity\_at TIMESTAMP DEFAULT NOW(),

-- SEO and Search

search\_keywords JSONB DEFAULT '[]',

tags JSONB DEFAULT '[]',

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

-- Job Applications

CREATE TABLE job\_applications (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

job\_posting\_id UUID REFERENCES job\_postings(id) ON DELETE CASCADE,

job\_seeker\_id UUID REFERENCES job\_seeker\_profiles(id) ON DELETE CASCADE,

-- Application Content

cover\_message TEXT,

voice\_message\_url VARCHAR(500),

voice\_message\_duration INTEGER, -- in seconds

resume\_url VARCHAR(500),

portfolio\_urls JSONB DEFAULT '[]',

-- Application Details

expected\_salary DECIMAL(10,2),

salary\_negotiable BOOLEAN DEFAULT TRUE,

availability\_start\_date DATE,

notice\_period\_days INTEGER DEFAULT 0,

willing\_to\_relocate BOOLEAN DEFAULT FALSE,

transport\_arrangement VARCHAR(100), -- own\_vehicle, public\_transport, company\_transport

-- Questionnaire Responses

screening\_questions\_responses JSONB DEFAULT '{}',

skill\_assessments JSONB DEFAULT '{}',

availability\_details JSONB DEFAULT '{}',

-- Status Tracking

status application\_status\_enum DEFAULT 'applied',

employer\_notes TEXT,

rejection\_reason VARCHAR(500),

rejection\_feedback TEXT,

-- Interview Information

interview\_scheduled\_at TIMESTAMP,

interview\_type interview\_type\_enum,

interview\_location\_address JSONB,

interview\_contact\_person VARCHAR(255),

interview\_contact\_phone VARCHAR(15),

interview\_instructions TEXT,

interview\_completed\_at TIMESTAMP,

interview\_rating INTEGER, -- 1-10

interview\_feedback TEXT,

interview\_notes TEXT,

-- Selection and Hiring

selected\_at TIMESTAMP,

job\_offer\_details JSONB DEFAULT '{}',

offer\_accepted\_at TIMESTAMP,

offer\_rejected\_at TIMESTAMP,

rejection\_by\_candidate\_reason TEXT,

-- Work Commencement

work\_start\_date DATE,

work\_started\_at TIMESTAMP,

probation\_period\_days INTEGER,

-- Communication Log

last\_employer\_message\_at TIMESTAMP,

last\_candidate\_message\_at TIMESTAMP,

unread\_messages\_employer INTEGER DEFAULT 0,

unread\_messages\_candidate INTEGER DEFAULT 0,

-- Metadata

applied\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW(),

source VARCHAR(50) DEFAULT 'mobile\_app', -- mobile\_app, website, referral, agent

UNIQUE(job\_posting\_id, job\_seeker\_id)

);

-- Saved Jobs (Bookmarks)

CREATE TABLE saved\_jobs (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

job\_seeker\_id UUID REFERENCES job\_seeker\_profiles(id) ON DELETE CASCADE,

job\_posting\_id UUID REFERENCES job\_postings(id) ON DELETE CASCADE,

notes TEXT,

reminder\_date DATE,

reminder\_sent BOOLEAN DEFAULT FALSE,

saved\_at TIMESTAMP DEFAULT NOW(),

UNIQUE(job\_seeker\_id, job\_posting\_id)

);

-- Indexes for job system

CREATE INDEX CONCURRENTLY idx\_job\_postings\_employer ON job\_postings(employer\_id);

CREATE INDEX CONCURRENTLY idx\_job\_postings\_category ON job\_postings(category\_id);

CREATE INDEX CONCURRENTLY idx\_job\_postings\_location ON job\_postings USING GIST(job\_location);

CREATE INDEX CONCURRENTLY idx\_job\_postings\_status ON job\_postings(status) WHERE status = 'active';

CREATE INDEX CONCURRENTLY idx\_job\_postings\_urgency ON job\_postings(urgency, created\_at DESC) WHERE status = 'active';

CREATE INDEX CONCURRENTLY idx\_job\_postings\_salary ON job\_postings(salary\_min, salary\_max) WHERE status = 'active';

CREATE INDEX CONCURRENTLY idx\_job\_postings\_search ON job\_postings USING GIN(to\_tsvector('english', title || ' ' || description));

CREATE INDEX CONCURRENTLY idx\_job\_postings\_expires ON job\_postings(expires\_at) WHERE status = 'active';

CREATE INDEX CONCURRENTLY idx\_job\_applications\_job ON job\_applications(job\_posting\_id);

CREATE INDEX CONCURRENTLY idx\_job\_applications\_seeker ON job\_applications(job\_seeker\_id);

CREATE INDEX CONCURRENTLY idx\_job\_applications\_status ON job\_applications(status, applied\_at DESC);

CREATE INDEX CONCURRENTLY idx\_job\_applications\_interview ON job\_applications(interview\_scheduled\_at) WHERE interview\_scheduled\_at IS NOT NULL;

CREATE INDEX CONCURRENTLY idx\_saved\_jobs\_seeker ON saved\_jobs(job\_seeker\_id, saved\_at DESC);

**4.2 MongoDB Collections for Document Storage**

**4.2.1 User Documents Collection**

// User Documents Collection - Stores uploaded documents

{

\_id: ObjectId(),

userId: "uuid", // Reference to users table

profileId: "uuid", // Reference to job\_seeker\_profiles or employer\_profiles

// Document Information

documentType: "aadhaar|pan|driving\_license|educational\_certificate|work\_certificate|business\_license|gst\_certificate",

category: "identity|education|work\_experience|business\_registration|address\_proof",

// File Information

originalName: "aadhaar\_card.pdf",

fileName: "doc\_uuid\_encrypted.pdf", // Encrypted filename for security

fileUrl: "https://storage.example.com/documents/doc\_uuid\_encrypted.pdf",

thumbnailUrl: "https://storage.example.com/thumbnails/doc\_uuid\_thumb.jpg",

mimeType: "application/pdf",

fileSize: 1234567, // in bytes

fileSizeFormatted: "1.2 MB",

// Upload Information

uploadedVia: "mobile\_app|website|api",

uploadLocation: {

country: "India",

state: "Karnataka",

city: "Bangalore",

coordinates: [77.5946, 12.9716]

},

// Verification Details

verificationStatus: "pending|in\_progress|verified|rejected|expired",

verificationDetails: {

method: "ocr|manual|api", // How it was verified

confidence: 0.95, // 0.0 to 1.0

ocrText: "extracted text from document",

extractedData: {

name: "John Doe",

documentNumber: "1234-5678-9012",

issueDate: "2020-01-01",

expiryDate: "2030-01-01",

address: "123 Main St, City, State, PIN",

dateOfBirth: "1990-01-01",

fatherName: "Father Name"

},

verificationDate: ISODate("2024-01-15T10:30:00Z"),

verifiedBy: "uuid", // User ID of verifier

verificationNotes: "Document appears authentic",

rejectionReason: "Unclear image quality"

},

// Processing Status

processingStatus: "uploaded|processing|processed|failed",

processingLog: [

{

step: "virus\_scan",

status: "completed",

timestamp: ISODate("2024-01-15T10:30:00Z"),

details: "No threats detected"

},

{

step: "ocr\_extraction",

status: "completed",

timestamp: ISODate("2024-01-15T10:31:00Z"),

details: "Text extraction completed with 95% confidence"

}

],

// Metadata

tags: ["identity", "primary", "government\_issued"],

expiryDate: ISODate("2030-01-01T00:00:00Z"),

isExpired: false,

isPrimary: true, // Primary document for this category

// Privacy and Compliance

retentionPolicy: "7\_years", // How long to keep the document

gdprConsent: {

processing: true,

storage: true,

sharing: false,

consentDate: ISODate("2024-01-15T10:25:00Z")

},

uploadedAt: ISODate("2024-01-15T10:30:00Z"),

updatedAt: ISODate("2024-01-15T10:35:00Z")

}

**4.2.2 Job Analytics Collection**

// Job Performance Analytics Collection

{

\_id: ObjectId(),

jobPostingId: "uuid", // Reference to job\_postings table

employerId: "uuid", // Reference to employer\_profiles table

// Time Period

analyticsPeriod: "daily|weekly|monthly",

startDate: ISODate("2024-01-01T00:00:00Z"),

endDate: ISODate("2024-01-31T23:59:59Z"),

// View Analytics

viewsData: {

totalViews: 1250,

uniqueViews: 980,

averageViewDuration: 45, // seconds

bounceRate: 0.35, // percentage of single-page sessions

// Daily breakdown

dailyViews: [

{

date: "2024-01-01",

views: 45,

uniqueViews: 38,

averageDuration: 42,

topViewingHours: [9, 10, 14, 15, 18, 19] // Peak hours

}

],

// Geographic distribution

viewsByLocation: [

{

city: "Bangalore",

state: "Karnataka",

views: 345,

uniqueViews: 287

},

{

city: "Mumbai",

state: "Maharashtra",

views: 198,

uniqueViews: 165

}

],

// Device and platform analytics

deviceBreakdown: {

mobile: { count: 892, percentage: 71.4 },

desktop: { count: 245, percentage: 19.6 },

tablet: { count: 113, percentage: 9.0 }

}

},

// Application Analytics

applicationsData: {

totalApplications: 87,

qualifiedApplications: 45,

shortlistedApplications: 12,

interviewedCandidates: 8,

selectedCandidates: 2,

// Conversion funnel

conversionRates: {

viewToApplication: 0.0696, // 87/1250

applicationToShortlist: 0.138, // 12/87

shortlistToInterview: 0.667, // 8/12

interviewToSelection: 0.25 // 2/8

},

// Applications by skills

applicationsBySkill: {

"masonry": { count: 35, qualified: 20, selected: 1 },

"construction": { count: 52, qualified: 25, selected: 1 },

"carpentry": { count: 15, qualified: 8, selected: 0 }

},

// Geographic distribution of applicants

applicationsByLocation: {

"within\_5km": { count: 45, percentage: 51.7 },

"5\_to\_10km": { count: 28, percentage: 32.2 },

"10\_to\_20km": { count: 12, percentage: 13.8 },

"above\_20km": { count: 2, percentage: 2.3 }

}

},

// Performance Metrics

performanceMetrics: {

averageResponseTime: "2.5 days", // Time to respond to applications

responseTimeHours: 60,

interviewConversionRate: 0.667,

hiringSuccessRate: 0.25,

timeToFill: 18, // days from posting to hiring

costPerHire: 250, // in currency units

qualityOfHire: 4.2 // based on subsequent performance/reviews

},

lastUpdated: ISODate("2024-02-01T00:00:00Z"),

dataSource: "automated\_collection",

version: "1.0"

}

**4.3 Redis Data Structures and Caching Strategy**

**4.3.1 User Session Management**

# User sessions with detailed information

HSET user:session:{sessionId}

userId "uuid"

userType "job\_seeker"

loginTime "2024-01-15T10:30:00Z"

lastActivity "2024-01-15T14:25:00Z"

deviceInfo '{"platform":"android","model":"Samsung Galaxy A12","appVersion":"1.2.3"}'

location '{"city":"Bangalore","state":"Karnataka","coordinates":[77.5946,12.9716]}'

permissions '["location","camera","notifications"]'

EXPIRE user:session:{sessionId} 86400 # 24 hours

# Active user tracking

SADD online\_users {userId}

EXPIRE online\_users 300 # 5 minutes

# User preferences cache

HSET user:preferences:{userId}

language "hi"

notifications\_enabled "true"

job\_alert\_radius "15"

salary\_alerts "true"

push\_notifications "true"

sms\_notifications "false"

EXPIRE user:preferences:{userId} 7200 # 2 hours

**4.3.2 Job Search and Matching Cache**

# Job search results cache with complex key generation

SET jobs:search:{hashOfQueryParams} '[

{"id":"job\_uuid\_1","title":"Construction Worker","location":"Bangalore","salary":18000},

{"id":"job\_uuid\_2","title":"Mason Required","location":"Bangalore","salary":20000}

]'

EXPIRE jobs:search:{hashOfQueryParams} 1800 # 30 minutes

# Popular searches cache

ZINCRBY popular\_searches 1 "construction worker bangalore"

ZINCRBY popular\_searches 1 "delivery boy mumbai"

# Job recommendations for users (personalized)

SET user:recommendations:{userId} '[

{"jobId":"uuid1","score":0.95,"reason":"skill\_match"},

{"jobId":"uuid2","score":0.87,"reason":"location\_proximity"}

]'

EXPIRE user:recommendations:{userId} 3600 # 1 hour

# Job view counts (real-time)

HINCRBY job:stats:{jobId} views\_count 1

HINCRBY job:stats:{jobId} applications\_count 1

**4.3.3 Application Rate Limiting and Security**

# API rate limiting by user

INCR api:rate\_limit:{userId}:job\_search

EXPIRE api:rate\_limit:{userId}:job\_search 3600 # Reset every hour

# OTP rate limiting

INCR otp:attempts:{phoneNumber}

EXPIRE otp:attempts:{phoneNumber} 900 # 15 minutes

# Failed login attempt tracking

INCR login:failed:{phoneNumber}

EXPIRE login:failed:{phoneNumber} 1800 # 30 minutes

"responseRate": 0.89, "avgResponseTime": "4 hours", "logo": "https://cdn.jobportal.com/logos/abc\_construction.jpg", "website": "https://abcconstruction.com", "location": { "city": "Bangalore", "state": "Karnataka" }, "otherActiveJobs": 12, "totalHires": 89, "hiringSuccessRate": 0.78 }, "metrics": { "applicationsCount": 23, "viewsCount": 156, "uniqueViewsCount": 134, "shortlistCount": 5, "averageApplicationTime": "2.5 days", "competitionLevel": "moderate" }, "timeline": { "postedDate": "2024-01-15T10:30:00Z", "expiresAt": "2024-02-15T23:59:59Z", "startDate": "2024-01-20", "expectedDuration": "3 months", "urgency": "immediate" }, "insights": { "similarJobsCount": 15, "averageSalaryInArea": 950, "demandTrend": "increasing", "bestTimeToApply": "morning", "successRate": "68% of applications get response" }, "userContext": { "matchScore": 0.94, "canApply": true, "applicationStatus": null, "missingRequirements": [], "strengthAreas": ["skill\_match", "location\_preference"], "recommendationReason": "Perfect match for your masonry skills and location preference" } } } }

### 4.2 Job Posting APIs (Employer)

#### List Jobs

```http

GET /jobs

Authorization: Bearer {token}

Query Parameters: ?status=active&category=construction&limit=20&sort=created\_at:desc

Response:

{

"success": true,

"data": {

"jobs": [

{

"id": "job\_550e8400",

"title": "Experienced Mason Required",

"status": "active",

"category": {

"id": "construction",

"name": "Construction"

},

"location": {

"city": "Bangalore",

"state": "Karnataka"

},

"salary": {

"min": 800,

"max": 1200,

"type": "daily"

},

"urgency": "immediate",

"metrics": {

"applicationsCount": 23,

"viewsCount": 156,

"shortlistCount": 5,

"responseRate": 0.78

},

"dates": {

"postedAt": "2024-01-15T10:30:00Z",

"expiresAt": "2024-02-15T23:59:59Z",

"lastActivity": "2024-01-16T14:20:00Z"

},

"featured": false,

"premium": true,

"boostLevel": 0

}

],

"summary": {

"totalJobs": 45,

"activeJobs": 12,

"filledJobs": 28,

"expiredJobs": 5,

"totalApplications": 456,

"pendingApplications": 34

}

}

}

**Create Job**

POST /jobs

Authorization: Bearer {token}

Content-Type: application/json

{

"basicInfo": {

"title": "Experienced Mason Required - Residential Project",

"description": "We are looking for an experienced mason for our residential construction project...",

"shortDescription": "Skilled mason needed for 3-month residential project in Whitefield",

"categoryId": "construction",

"jobType": "full\_time"

},

"skills": {

"required": [

{

"skillId": "masonry",

"experienceLevel": "intermediate",

"minExperience": 2,

"essential": true

}

],

"optional": [

{

"skillId": "concrete\_work",

"experienceLevel": "beginner"

}

],

"flexibleRequirements": true

},

"location": {

"coordinates": [77.7500, 12.9698],

"address": {

"line1": "Whitefield Main Road",

"line2": "Near ITPL",

"city": "Bangalore",

"state": "Karnataka",

"pincode": "560066",

"landmark": "Opposite Forum Mall"

},

"description": "Easy access by public transport",

"searchRadius": 15,

"remoteWorkAllowed": false,

"onSiteRequired": true,

"travelRequired": false

},

"compensation": {

"paymentType": "daily",

"salaryMin": 800,

"salaryMax": 1200,

"currency": "INR",

"negotiable": false,

"benefits": [

"food\_provided",

"transport\_allowance",

"overtime\_pay"

],

"bonusDetails": "Performance bonus after project completion",

"paymentTerms": "Weekly payment on Saturdays"

},

"requirements": {

"experienceLevel": "intermediate",

"minAge": 21,

"maxAge": 45,

"genderPreference": null,

"educationRequirements": "Primary education preferred",

"physicalRequirements": "Ability to lift 25kg, work at heights",

"additionalRequirements": "Should have own basic tools",

"languageRequirements": ["hindi", "kannada"],

"certifications": ["safety\_training\_preferred"]

},

"timeline": {

"urgency": "immediate",

"startDate": "2024-01-22",

"endDate": "2024-04-22",

"durationType": "months",

"durationValue": 3,

"workingHours": {

"start": "08:00",

"end": "17:00",

"breakMinutes": 60

},

"workingDays": ["monday", "tuesday", "wednesday", "thursday", "friday", "saturday"],

"shiftDetails": "Single day shift only"

},

"applicationSettings": {

"applicationDeadline": "2024-01-25T23:59:59Z",

"instructions": "Please bring work samples and references",

"contactPerson": "Rajesh Kumar",

"contactPhone": "+919876543210",

"contactEmail": "rajesh@abcconstruction.com",

"screeningQuestions": [

{

"question": "How many years of masonry experience do you have?",

"type": "number",

"required": true

},

{

"question": "Can you work at heights?",

"type": "boolean",

"required": true

}

]

},

"additionalInfo": {

"tags": ["residential", "immediate\_start", "long\_term"],

"searchKeywords": ["mason", "construction", "building", "brick\_work"]

}

}

Response:

{

"success": true,

"data": {

"job": {

"id": "job\_new\_550e8400",

"title": "Experienced Mason Required - Residential Project",

"status": "draft",

"employer": {

"id": "emp\_123",

"name": "ABC Construction Pvt Ltd"

},

"createdAt": "2024-01-16T16:30:00Z",

"estimatedReach": 1250,

"suggestedPromotions": [

{

"type": "featured",

"cost": 500,

"benefit": "3x more visibility"

},

{

"type": "urgent",

"cost": 200,

"benefit": "Appears at top of search"

}

]

},

"nextSteps": [

{

"step": "review\_job",

"description": "Review job details before publishing",

"action": "GET /jobs/{jobId}/preview"

},

{

"step": "publish\_job",

"description": "Publish job to make it visible to candidates",

"action": "POST /jobs/{jobId}/publish"

}

]

},

"message": "Job created successfully. Review and publish to make it live."

}

**Update Job**

PUT /jobs/550e8400-e29b-41d4-a716-446655440000

Authorization: Bearer {token}

Content-Type: application/json

{

"basicInfo": {

"title": "Senior Mason Required - Residential Project",

"description": "Updated job description..."

},

"compensation": {

"salaryMin": 900,

"salaryMax": 1300

},

"timeline": {

"applicationDeadline": "2024-01-30T23:59:59Z"

}

}

Response:

{

"success": true,

"data": {

"job": {

// Updated job object

},

"updatedFields": [

"basicInfo.title",

"compensation.salaryMin",

"compensation.salaryMax",

"timeline.applicationDeadline"

],

"republished": false,

"notificationsSent": 15

}

}

**Publish Job**

POST /jobs/550e8400-e29b-41d4-a716-446655440000/publish

Authorization: Bearer {token}

Response:

{

"success": true,

"data": {

"job": {

"id": "job\_550e8400",

"status": "active",

"publishedAt": "2024-01-16T17:00:00Z",

"expiresAt": "2024-02-16T17:00:00Z"

},

"reach": {

"estimatedCandidates": 1250,

"notificationsSent": 89,

"eligibleCandidates": 456

},

"visibility": {

"searchRanking": "high",

"featuredStatus": false,

"boostLevel": 0

}

}

}

**Pause Job**

POST /jobs/550e8400-e29b-41d4-a716-446655440000/pause

Authorization: Bearer {token}

Content-Type: application/json

{

"reason": "Taking a break for review",

"resumeAt": "2024-01-20T09:00:00Z"

}

Response:

{

"success": true,

"data": {

"job": {

"id": "job\_550e8400",

"status": "paused",

"pausedAt": "2024-01-16T17:30:00Z",

"resumeAt": "2024-01-20T09:00:00Z"

},

"impact": {

"hiddenFromSearch": true,

"applicationsDisabled": true,

"existingApplicationsRetained": true

}

}

}

**5. API Architecture**

**5.1 RESTful API Design**

**Base URL Structure**

Production: https://api.jobportal.com/v1

Staging: https://api-staging.jobportal.com/v1

Development: https://api-dev.jobportal.com/v1

**Authentication Endpoints**

yaml

POST /auth/register

POST /auth/verify-otp

POST /auth/login

POST /auth/refresh-token

POST /auth/logout

POST /auth/forgot-password

POST /auth/reset-password

**User Management APIs**

yaml

*# Job Seekers*

GET /job-seekers/profile

PUT /job-seekers/profile

POST /job-seekers/profile/photo

GET /job-seekers/skills

POST /job-seekers/skills

DELETE /job-seekers/skills/{skillId}

GET /job-seekers/documents

POST /job-seekers/documents

DELETE /job-seekers/documents/{documentId}

*# Employers*

GET /employers/profile

PUT /employers/profile

GET /employers/verification-status

POST /employers/documents

**Job Management APIs**

yaml

*# Job Postings (Employer)*

GET /jobs *# List employer's jobs*

POST /jobs *# Create new job*

GET /jobs/{jobId} *# Get specific job*

PUT /jobs/{jobId} *# Update job*

DELETE /jobs/{jobId} *# Delete job*

POST /jobs/{jobId}/duplicate *# Duplicate existing job*

*# Job Applications (Employer view)*

GET /jobs/{jobId}/applications

GET /jobs/{jobId}/applications/{applicationId}

PUT /jobs/{jobId}/applications/{applicationId}/status

POST /jobs/{jobId}/applications/{applicationId}/interview

*# Job Search (Job Seeker)*

GET /search/jobs *# Search jobs with filters*

GET /search/jobs/nearby *# Location-based search*

GET /search/jobs/recommended *# AI-recommended jobs*

GET /jobs/{jobId}/details *# Public job details*

*# Job Applications (Job Seeker)*

POST /jobs/{jobId}/apply *# Apply for job*

GET /applications *# List user's applications*

GET /applications/{applicationId}

PUT /applications/{applicationId}/withdraw

**5.2 API Request/Response Examples**

**Job Search API**

http

GET /v1/search/jobs?location=28.7041,77.1025&radius=10&category=construction&salary\_min=500&salary\_max=1500&page=1&limit=20

Response:

{

"success": true,

"data": {

"jobs": [

{

"id": "job-uuid-1",

"title": "Construction Worker Needed",

"description": "Looking for experienced construction worker for residential project",

"category": {

"id": "construction",

"name": "Construction",

"name\_hindi": "निर्माण"

},

"skills": [

{"id": "masonry", "name": "Masonry", "required": true},

{"id": "concrete", "name": "Concrete Work", "required": false}

],

"location": {

"coordinates": [77.1025, 28.7041],

"address": "Sector 15, Gurgaon, Haryana",

"distance": 2.5

},

"salary": {

"type": "daily",

"min": 800,

"max": 1200,

"currency": "INR"

},

"urgency": "immediate",

"employer": {

"name": "ABC Construction",

"rating": 4.2,

"verified": true

},

"posted\_date": "2024-01-15T10:30:00Z",

"applications\_count": 5,

"expires\_at": "2024-02-15T10:30:00Z"

}

],

"pagination": {

"page": 1,

"limit": 20,

"total": 150,

"total\_pages": 8,

"has\_next": true,

"has\_previous": false

},

"filters\_applied": {

"location": "28.7041,77.1025",

"radius": 10,

"category": "construction",

"salary\_range": [500, 1500]

}

},

"message": "Jobs retrieved successfully"

}

**Job Application API**

http

POST /v1/jobs/job-uuid-1/apply

Content-Type: application/json

{

"cover\_message": "I have 3 years of experience in construction work",

"voice\_message\_url": "https://storage../voice-messages/msg-uuid.mp3",

"expected\_salary": 1000,

"availability\_start\_date": "2024-01-20"

}

Response:

{

"success": true,

"data": {

"application\_id": "application-uuid",

"job\_id": "job-uuid-1",

"status": "applied",

"applied\_at": "2024-01-16T14:30:00Z",

"message": "Your application has been submitted successfully"

}

}

**5.3 GraphQL Schema (for complex queries)**

graphql

type JobSeeker {

id: ID!

profile: JobSeekerProfile!

skills: [JobSeekerSkill!]!

applications: [JobApplication!]!

documents: [Document!]!

verificationStatus: VerificationStatus!

}

type JobPosting {

id: ID!

title: String!

description: String!

category: SkillCategory!

requiredSkills: [SkillRequirement!]!

location: Location!

salary: SalaryDetails!

urgency: Urgency!

employer: Employer!

applications(status: ApplicationStatus): [JobApplication!]!

applicationsCount: Int!

viewsCount: Int!

postedDate: DateTime!

expiresAt: DateTime

}

type Query {

jobSeeker(id: ID!): JobSeeker

jobPostings(

filters: JobFilters

location: LocationInput

pagination: PaginationInput

): JobPostingConnection!

*# Recommended jobs based on user profile*

recommendedJobs(

jobSeekerId: ID!

limit: Int = 10

): [JobPosting!]!

*# Job matching score*

jobMatchScore(

jobSeekerId: ID!

jobId: ID!

): JobMatchResult!

}

type Mutation {

*# Job seeker mutations*

updateJobSeekerProfile(input: JobSeekerProfileInput!): JobSeeker!

addSkill(input: AddSkillInput!): JobSeekerSkill!

applyForJob(input: JobApplicationInput!): JobApplication!

*# Employer mutations*

createJobPosting(input: JobPostingInput!): JobPosting!

updateApplicationStatus(

applicationId: ID!

status: ApplicationStatus!

notes: String

): JobApplication!

}

**5.4 API Rate Limiting Strategy**

yaml

*# Rate limits by user type*

Job Seekers:

- Search API: 100 requests/hour

- Apply API: 20 applications/day

- Profile Updates: 10 updates/day

- File Uploads: 50 files/day (max 5MB each)

Employers:

- Job Posting: 20 jobs/day

- Applications View: 500 requests/hour

- Bulk Actions: 100 actions/hour

- File Downloads: 200 files/hour

Public APIs:

- Job Search (unauthenticated): 50 requests/hour/IP

- Registration: 5 attempts/hour/IP

- OTP Requests: 3 requests/5minutes/phone

Premium Features:

- Enhanced Search: 500 requests/hour

- Priority Support: Unlimited

- Advanced Analytics: 1000 requests/hour

**6. Mobile Application Architecture**

**6.1 React Native Architecture**

**Project Structure**

src/

├── components/ # Reusable UI components

│ ├── common/ # Generic components (Button, Input, etc.)

│ ├── forms/ # Form-specific components

│ └── job/ # Job-related components

├── screens/ # Screen components

│ ├── auth/ # Authentication screens

│ ├── jobs/ # Job-related screens

│ ├── profile/ # Profile management screens

│ └── messaging/ # Chat and messaging screens

├── navigation/ # Navigation configuration

├── services/ # API services and business logic

│ ├── api/ # API client and endpoints

│ ├── auth/ # Authentication service

│ ├── storage/ # Local storage service

│ └── location/ # Location services

├── store/ # Redux store configuration

│ ├── slices/ # Redux Toolkit slices

│ └── middleware/ # Custom middleware

├── utils/ # Utility functions

├── constants/ # App constants and configurations

├── hooks/ # Custom React hooks

├── types/ # TypeScript type definitions

└── assets/ # Images, fonts, etc.

**State Management with Redux Toolkit**

// store/slices/authSlice.ts

import { createSlice, createAsyncThunk } from '@reduxjs/toolkit'

interface AuthState {

user: User | null

token: string | null

isLoading: boolean

error: string | null

isFirstTimeUser: boolean

}

export const loginWithOTP = createAsyncThunk(

'auth/loginWithOTP',

async (credentials: OTPCredentials) => {

const response = await authAPI.verifyOTP(credentials)

await SecureStore.setItemAsync('token', response.data.token)

return response.data

}

)

const authSlice = createSlice({

name: 'auth',

initialState,

reducers: {

logout: (state) => {

state.user = null

state.token = null

SecureStore.deleteItemAsync('token')

},

clearError: (state) => {

state.error = null

}

},

extraReducers: (builder) => {

builder

.addCase(loginWithOTP.pending, (state) => {

state.isLoading = true

state.error = null

})

.addCase(loginWithOTP.fulfilled, (state, action) => {

state.isLoading = false

state.user = action.payload.user

state.token = action.payload.token

})

.addCase(loginWithOTP.rejected, (state, action) => {

state.isLoading = false

state.error = action.error.message || 'Login failed'

})

}

})

**Offline-First Architecture**

// services/storage/OfflineManager.ts

import SQLite from 'react-native-sqlite-storage'

import NetInfo from '@react-native-community/netinfo'

import AsyncStorage from '@react-native-async-storage/async-storage'

class OfflineManager {

private db: SQLite.SQLiteDatabase

private syncQueue: SyncOperation[] = []

private isOnline: boolean = false

async initialize() {

this.db = await SQLite.openDatabase({

name: 'jobportal\_offline.db',

location: 'default'

})

await this.createTables()

this.setupNetworkListener()

await this.loadSyncQueue()

}

private async createTables() {

const tables = [

`CREATE TABLE IF NOT EXISTS cached\_jobs (

id TEXT PRIMARY KEY,

data TEXT NOT NULL,

cached\_at INTEGER NOT NULL,

expires\_at INTEGER NOT NULL

)`,

`CREATE TABLE IF NOT EXISTS draft\_applications (

id TEXT PRIMARY KEY,

job\_id TEXT NOT NULL,

data TEXT NOT NULL,

created\_at INTEGER NOT NULL

)`,

`CREATE TABLE IF NOT EXISTS sync\_queue (

id TEXT PRIMARY KEY,

operation\_type TEXT NOT NULL,

data TEXT NOT NULL,

created\_at INTEGER NOT NULL,

retry\_count INTEGER DEFAULT 0

)`

]

for (const table of tables) {

await this.db.executeSql(table)

}

}

async cacheJobData(jobs: Job[], expiryHours: number = 24) {

const expiresAt = Date.now() + (expiryHours \* 60 \* 60 \* 1000)

for (const job of jobs) {

await this.db.executeSql(

'INSERT OR REPLACE INTO cached\_jobs (id, data, cached\_at, expires\_at) VALUES (?, ?, ?, ?)',

[job.id, JSON.stringify(job), Date.now(), expiresAt]

)

}

}

async getCachedJobs(): Promise<Job[]> {

const [results] = await this.db.executeSql(

'SELECT data FROM cached\_jobs WHERE expires\_at > ? ORDER BY cached\_at DESC',

[Date.now()]

)

return results.rows.raw().map(row => JSON.parse(row.data))

}

async queueForSync(operation: SyncOperation) {

this.syncQueue.push(operation)

await this.saveSyncQueue()

if (this.isOnline) {

await this.processSyncQueue()

}

}

private async processSyncQueue() {

while (this.syncQueue.length > 0 && this.isOnline) {

const operation = this.syncQueue[0]

try {

await this.executeOperation(operation)

this.syncQueue.shift()

await this.saveSyncQueue()

} catch (error) {

operation.retry\_count++

if (operation.retry\_count >= 3) {

this.syncQueue.shift() // Remove failed operation

}

break // Stop processing on error

}

}

}

}

**Voice Integration Service**

// services/voice/VoiceService.ts

import Voice from '@react-native-voice/voice'

import Tts from 'react-native-tts'

import RNFS from 'react-native-fs'

class VoiceService {

private isListening: boolean = false

private currentLanguage: string = 'hi-IN'

async initializeVoiceServices() {

Voice.onSpeechStart = this.onSpeechStart

Voice.onSpeechEnd = this.onSpeechEnd

Voice.onSpeechResults = this.onSpeechResults

Voice.onSpeechError = this.onSpeechError

// Initialize TTS

await Tts.setDefaultLanguage(this.currentLanguage)

await Tts.setDefaultRate(0.5)

await Tts.setDefaultPitch(1.0)

}

async startListening(language: string = 'hi-IN') {

try {

this.currentLanguage = language

await Voice.start(language)

this.isListening = true

} catch (error) {

console.error('Voice recognition error:', error)

}

}

async stopListening() {

try {

await Voice.stop()

this.isListening = false

} catch (error) {

console.error('Stop listening error:', error)

}

}

async speakText(text: string, language?: string) {

try {

if (language && language !== this.currentLanguage) {

await Tts.setDefaultLanguage(language)

this.currentLanguage = language

}

await Tts.speak(text)

} catch (error) {

console.error('TTS error:', error)

}

}

async recordVoiceMessage(maxDuration: number = 60000): Promise<string> {

const audioPath = `${RNFS.CachesDirectoryPath}/voice\_message\_${Date.now()}.mp3`

// Implementation would use react-native-audio-recorder-player

// Return the file path after recording

return audioPath

}

private onSpeechResults = (event: any) => {

const results = event.value

if (results && results.length > 0) {

// Emit speech recognition results

this.emitSpeechResult(results[0])

}

}

}

**6.2 Performance Optimization Strategies**

**Image Optimization**

// components/common/OptimizedImage.tsx

import React, { useState } from 'react'

import { Image, ImageStyle, ActivityIndicator } from 'react-native'

import FastImage from 'react-native-fast-image'

interface OptimizedImageProps {

source: { uri: string }

style?: ImageStyle

placeholder?: string

quality?: 'low' | 'medium' | 'high'

}

const OptimizedImage: React.FC<OptimizedImageProps> = ({

source,

style,

placeholder,

quality = 'medium'

}) => {

const [loading, setLoading] = useState(true)

const [error, setError] = useState(false)

const getOptimizedUrl = (url: string, quality: string) => {

// Add query parameters for image optimization

const params = new URLSearchParams({

w: '400', // max width

h: '400', // max height

q: quality === 'low' ? '60' : quality === 'medium' ? '75' : '90',

f: 'webp' // preferred format

})

return `${url}?${params.toString()}`

}

return (

<FastImage

source={{

uri: getOptimizedUrl(source.uri, quality),

priority: FastImage.priority.normal

}}

style={style}

onLoadStart={() => setLoading(true)}

onLoadEnd={() => setLoading(false)}

onError={() => setError(true)}

resizeMode={FastImage.resizeMode.cover}

/>

)

}

**Memory Management**

// utils/MemoryManager.ts

class MemoryManager {

private imageCache = new Map<string, any>()

private maxCacheSize = 50 // Maximum cached images

cacheImage(key: string, image: any) {

if (this.imageCache.size >= this.maxCacheSize) {

// Remove oldest cached image

const firstKey = this.imageCache.keys().next().value

this.imageCache.delete(firstKey)

}

this.imageCache.set(key, image)

}

getCachedImage(key: string) {

return this.imageCache.get(key)

}

clearCache() {

this.imageCache.clear()

}

// Cleanup on memory warning

onMemoryWarning() {

this.clearCache()

// Additional cleanup operations

}

}

**7. Backend Services**

**7.1 Microservices Architecture**

**Authentication Service**

// services/auth/AuthService.ts

import jwt from 'jsonwebtoken'

import bcrypt from 'bcrypt'

import { createClient } from 'redis'

class AuthService {

private redis = createClient({ url: process.env.REDIS\_URL })

async sendOTP(phoneNumber: string): Promise<boolean> {

const otp = this.generateOTP()

const key = `otp:${phoneNumber}`

// Store OTP in Redis with 5-minute expiry

await this.redis.setex(key, 300, otp)

// Send SMS via third-party service

await this.smsService.sendOTP(phoneNumber, otp)

return true

}

async verifyOTP(phoneNumber: string, otp: string): Promise<AuthResponse> {

const key = `otp:${phoneNumber}`

const storedOTP = await this.redis.get(key)

if (!storedOTP || storedOTP !== otp) {

throw new Error('Invalid or expired OTP')

}

// Delete used OTP

await this.redis.del(key)

// Find or create user

let user = await this.userRepository.findByPhone(phoneNumber)

if (!user) {

user = await this.userRepository.create({

phoneNumber,

status: 'pending'

})

}

// Generate tokens

const accessToken = this.generateAccessToken(user)

const refreshToken = this.generateRefreshToken(user)

// Store refresh token

await this.redis.setex(`refresh:${user.id}`, 2592000, refreshToken) // 30 days

return {

user,

accessToken,

refreshToken,

isFirstTimeUser: user.status === 'pending'

}

}

private generateAccessToken(user: User): string {

return jwt.sign(

{

userId: user.id,

userType: user.userType,

phoneNumber: user.phoneNumber

},

process.env.JWT\_SECRET!,

{ expiresIn: '1h' }

)

}

async validateToken(token: string): Promise<JWTPayload> {

try {

return jwt.verify(token, process.env.JWT\_SECRET!) as JWTPayload

} catch (error) {

throw new Error('Invalid token')

}

}

}

**Job Matching Service**

// services/jobs/JobMatchingService.ts

import { Client } from '@elastic/elasticsearch'

class JobMatchingService {

private elasticsearch = new Client({

node: process.env.ELASTICSEARCH\_URL

})

async findMatchingJobs(jobSeeker: JobSeeker, limit: number = 20): Promise<JobMatch[]> {

const userLocation = jobSeeker.currentLocation

const userSkills = jobSeeker.skills.map(s => s.skillId)

const searchQuery = {

index: 'job\_postings',

body: {

query: {

bool: {

must: [

{ term: { status: 'active' } },

{ range: { expires\_at: { gte: 'now' } } }

],

should: [

// Skill matching with boost

{

terms: {

'required\_skills.skill\_id': userSkills,

boost: 3.0

}

},

// Experience level matching

{

term: {

'experience\_required': jobSeeker.experienceLevel,

boost: 2.0

}

}

],

filter: [

// Geographic filtering

{

geo\_distance: {

distance: `${jobSeeker.preferredWorkRadius}km`,

location: {

lat: userLocation.latitude,

lon: userLocation.longitude

}

}

}

]

}

},

sort: [

'\_score',

{

'\_geo\_distance': {

'location': {

'lat': userLocation.latitude,

'lon': userLocation.longitude

},

'order': 'asc',

'unit': 'km'

}

}

],

size: limit

}

}

const response = await this.elasticsearch.search(searchQuery)

return response.body.hits.hits.map(hit => ({

job: hit.\_source,

matchScore: hit.\_score,

distance: hit.sort[1]

}))

}

async calculateJobScore(jobSeeker: JobSeeker, job: JobPosting): Promise<number> {

let score = 0

// Skill matching (40% weight)

const jobSkills = job.requiredSkills.map(rs => rs.skillId)

const userSkills = jobSeeker.skills.map(us => us.skillId)

const skillMatch = this.calculateSkillMatch(userSkills, jobSkills)

score += skillMatch \* 0.4

// Location proximity (30% weight)

const distance = this.calculateDistance(

jobSeeker.currentLocation,

job.location

)

const locationScore = Math.max(0, 1 - (distance / jobSeeker.preferredWorkRadius))

score += locationScore \* 0.3

// Salary alignment (20% weight)

const salaryScore = this.calculateSalaryScore(

jobSeeker.expectedSalaryRange,

job.salaryRange

)

score += salaryScore \* 0.2

// Employer rating (10% weight)

const employerScore = job.employer.rating / 5

score += employerScore \* 0.1

return Math.min(1, score) // Cap at 1.0

}

}

**Notification Service**

// services/notifications/NotificationService.ts

import admin from 'firebase-admin'

import { Queue, Worker } from 'bullmq'

class NotificationService {

private pushQueue: Queue

private smsQueue: Queue

constructor() {

this.pushQueue = new Queue('push-notifications', {

connection: { host: 'localhost', port: 6379 }

})

this.smsQueue = new Queue('sms-notifications', {

connection: { host: 'localhost', port: 6379 }

})

this.setupWorkers()

}

async sendJobMatchNotification(userId: string, job: JobPosting) {

const user = await this.userRepository.findById(userId)

// Send push notification

await this.pushQueue.add('job-match', {

userId,

title: 'New Job Match Found!',

body: `${job.title} - ${job.location.city}`,

data: {

jobId: job.id,

type: 'job\_match'

}

})

// Send SMS if enabled

if (user.preferences.smsNotifications) {

await this.smsQueue.add('job-match-sms', {

phoneNumber: user.phoneNumber,

message: `New job match: ${job.title} in ${job.location.city}. Apply now!`,

jobUrl: `https://app.jobportal.com/jobs/${job.id}`

})

}

}

private setupWorkers() {

// Push notification worker

new Worker('push-notifications', async (job) => {

const { userId, title, body, data } = job.data

const user = await this.userRepository.findById(userId)

if (!user.fcmToken) return

const message = {

token: user.fcmToken,

notification: { title, body },

data,

android: {

priority: 'high',

notification: {

sound: 'default',

clickAction: 'FLUTTER\_NOTIFICATION\_CLICK'

}

}

}

await admin.messaging().send(message)

})

// SMS worker

new Worker('sms-notifications', async (job) => {

const { phoneNumber, message } = job.data

await this.smsService.sendMessage(phoneNumber, message)

})

}

}

**7.2 File Upload & Processing Service**

// services/files/FileProcessingService.ts

import multer from 'multer'

import sharp from 'sharp'

import AWS from 'aws-sdk'

import { createWorker } from 'tesseract.js'

class FileProcessingService {

private s3: AWS.S3

private bucketName: string = process.env.AWS\_S3\_BUCKET!

constructor() {

this.s3 = new AWS.S3({

accessKeyId: process.env.AWS\_ACCESS\_KEY,

secretAccessKey: process.env.AWS\_SECRET\_KEY,

region: process.env.AWS\_REGION

})

}

// Configure multer for file uploads

getUploadMiddleware() {

const storage = multer.memoryStorage()

return multer({

storage,

limits: {

fileSize: 5 \* 1024 \* 1024, // 5MB limit

files: 5 // Max 5 files per request

},

fileFilter: (req, file, cb) => {

const allowedTypes = [

'image/jpeg', 'image/jpg', 'image/png',

'application/pdf', 'audio/mpeg', 'audio/wav'

]

if (allowedTypes.includes(file.mimetype)) {

cb(null, true)

} else {

cb(new Error('Invalid file type'))

}

}

})

}

async processAndUploadImage(

file: Express.Multer.File,

userId: string,

type: 'profile' | 'document' | 'work\_sample'

): Promise<FileUploadResult> {

// Process image with Sharp

const processedImage = await sharp(file.buffer)

.resize(800, 800, {

fit: 'inside',

withoutEnlargement: true

})

.jpeg({

quality: 80,

progressive: true

})

.toBuffer()

// Generate unique filename

const fileName = `${type}/${userId}/${Date.now()}-${file.originalname}`

// Upload to S3

const uploadResult = await this.s3.upload({

Bucket: this.bucketName,

Key: fileName,

Body: processedImage,

ContentType: 'image/jpeg',

ACL: 'private',

Metadata: {

userId,

fileType: type,

originalName: file.originalname,

uploadedAt: new Date().toISOString()

}

}).promise()

return {

fileUrl: uploadResult.Location,

fileName: fileName,

fileSize: processedImage.length,

mimeType: 'image/jpeg'

}

}

async extractTextFromDocument(fileBuffer: Buffer): Promise<string> {

const worker = await createWorker('eng+hin')

try {

const { data: { text } } = await worker.recognize(fileBuffer)

await worker.terminate()

return text

} catch (error) {

await worker.terminate()

throw new Error('OCR processing failed')

}

}

async generateSignedUrl(fileName: string, expiryMinutes: number = 60): Promise<string> {

const params = {

Bucket: this.bucketName,

Key: fileName,

Expires: expiryMinutes \* 60 // Convert to seconds

}

return this.s3.getSignedUrl('getObject', params)

}

}

**8. Third-Party Integrations**

**8.1 SMS Gateway Integration**

// services/integrations/SMSService.ts

import axios from 'axios'

interface SMSProvider {

sendOTP(phoneNumber: string, otp: string): Promise<boolean>

sendMessage(phoneNumber: string, message: string): Promise<boolean>

}

class TwilioSMSService implements SMSProvider {

private accountSid: string

private authToken: string

private fromNumber: string

constructor() {

this.accountSid = process.env.TWILIO\_ACCOUNT\_SID!

this.authToken = process.env.TWILIO\_AUTH\_TOKEN!

this.fromNumber = process.env.TWILIO\_FROM\_NUMBER!

}

async sendOTP(phoneNumber: string, otp: string): Promise<boolean> {

const message = `Your JobPortal verification code is: ${otp}. Valid for 5 minutes.`

return this.sendMessage(phoneNumber, message)

}

async sendMessage(phoneNumber: string, message: string): Promise<boolean> {

try {

const response = await axios.post(

`https://api.twilio.com/2010-04-01/Accounts/${this.accountSid}/Messages.json`,

{

From: this.fromNumber,

To: phoneNumber,

Body: message

},

{

auth: {

username: this.accountSid,

password: this.authToken

},

headers: {

'Content-Type': 'application/x-www-form-urlencoded'

}

}

)

return response.data.status !== 'failed'

} catch (error) {

console.error('SMS sending failed:', error)

return false

}

}

}

// Alternative Indian SMS provider

class MSG91SMSService implements SMSProvider {

private apiKey: string

private templateId: string

constructor() {

this.apiKey = process.env.MSG91\_API\_KEY!

this.templateId = process.env.MSG91\_TEMPLATE\_ID!

}

async sendOTP(phoneNumber: string, otp: string): Promise<boolean> {

try {

const response = await axios.post(

'https://api.msg91.com/api/v5/otp',

{

template\_id: this.templateId,

mobile: phoneNumber,

authkey: this.apiKey,

otp: otp,

otp\_expiry: 5 // minutes

}

)

return response.data.type === 'success'

} catch (error) {

console.error('MSG91 OTP failed:', error)

return false

}

}

async sendMessage(phoneNumber: string, message: string): Promise<boolean> {

try {

const response = await axios.post(

'https://api.msg91.com/api/v2/sendsms',

{

authkey: this.apiKey,

mobiles: phoneNumber,

message: message,

sender: 'JOBPRT',

route: 4

}

)

return response.data.type === 'success'

} catch (error) {

console.error('MSG91 SMS failed:', error)

return false

}

}

}

**8.2 Payment Gateway Integration**

// services/integrations/PaymentService.ts

import Razorpay from 'razorpay'

import crypto from 'crypto'

class PaymentService {

private razorpay: Razorpay

constructor() {

this.razorpay = new Razorpay({

key\_id: process.env.RAZORPAY\_KEY\_ID!,

key\_secret: process.env.RAZORPAY\_KEY\_SECRET!

})

}

async createPaymentOrder(

amount: number,

jobId: string,

employerId: string,

workerId: string

): Promise<PaymentOrder> {

const orderOptions = {

amount: amount \* 100, // Convert to paise

currency: 'INR',

receipt: `job\_${jobId}\_${Date.now()}`,

notes: {

jobId,

employerId,

workerId,

type: 'job\_payment'

}

}

const order = await this.razorpay.orders.create(orderOptions)

// Store order details in database

await this.paymentRepository.createOrder({

orderId: order.id,

amount,

jobId,

employerId,

workerId,

status: 'created'

})

return {

orderId: order.id,

amount: order.amount,

currency: order.currency,

keyId: process.env.RAZORPAY\_KEY\_ID!

}

}

async verifyPaymentSignature(

orderId: string,

paymentId: string,

signature: string

): Promise<boolean> {

const body = orderId + '|' + paymentId

const expectedSignature = crypto

.createHmac('sha256', process.env.RAZORPAY\_KEY\_SECRET!)

.update(body.toString())

.digest('hex')

return expectedSignature === signature

}

async processJobPayment(

orderId: string,

paymentId: string,

signature: string

): Promise<PaymentResult> {

// Verify signature

const isValid = await this.verifyPaymentSignature(orderId, paymentId, signature)

if (!isValid) {

throw new Error('Invalid payment signature')

}

// Get order details

const order = await this.paymentRepository.findByOrderId(orderId)

if (!order) {

throw new Error('Order not found')

}

// Calculate platform commission (5%)

const platformCommission = order.amount \* 0.05

const workerAmount = order.amount - platformCommission

// Process payment

const payment = await this.razorpay.payments.fetch(paymentId)

if (payment.status === 'captured') {

// Update payment status

await this.paymentRepository.updatePaymentStatus(orderId, 'completed')

// Transfer amount to worker (would integrate with bank transfer API)

await this.transferToWorker(order.workerId, workerAmount)

// Record commission

await this.recordPlatformCommission(order.jobId, platformCommission)

// Update job status

await this.jobRepository.updateJobStatus(order.jobId, 'completed')

return {

success: true,

paymentId,

workerAmount,

platformCommission

}

}

throw new Error('Payment not captured')

}

private async transferToWorker(workerId: string, amount: number) {

const worker = await this.userRepository.findById(workerId)

// Implementation would depend on chosen payout method

// Could be UPI, bank transfer, or wallet credit

if (worker.bankAccount) {

// Bank transfer implementation

await this.initiatesBankTransfer(worker.bankAccount, amount)

} else if (worker.upiId) {

// UPI transfer implementation

await this.initiateUPITransfer(worker.upiId, amount)

}

}

}

**8.3 Maps & Location Integration**

// services/integrations/LocationService.ts

import axios from 'axios'

class LocationService {

private googleMapsApiKey: string

constructor() {

this.googleMapsApiKey = process.env.GOOGLE\_MAPS\_API\_KEY!

}

async geocodeAddress(address: string): Promise<GeoLocation> {

try {

const response = await axios.get(

'https://maps.googleapis.com/maps/api/geocode/json',

{

params: {

address,

key: this.googleMapsApiKey,

region: 'in' // Bias towards India

}

}

)

const result = response.data.results[0]

if (!result) {

throw new Error('Address not found')

}

return {

latitude: result.geometry.location.lat,

longitude: result.geometry.location.lng,

formattedAddress: result.formatted\_address,

city: this.extractCity(result.address\_components),

state: this.extractState(result.address\_components),

pincode: this.extractPincode(result.address\_components)

}

} catch (error) {

throw new Error('Geocoding failed')

}

}

async reverseGeocode(latitude: number, longitude: number): Promise<AddressInfo> {

try {

const response = await axios.get(

'https://maps.googleapis.com/maps/api/geocode/json',

{

params: {

latlng: `${latitude},${longitude}`,

key: this.googleMapsApiKey

}

}

)

const result = response.data.results[0]

return {

formattedAddress: result.formatted\_address,

city: this.extractCity(result.address\_components),

state: this.extractState(result.address\_components),

pincode: this.extractPincode(result.address\_components),

landmark: this.extractLandmark(result.address\_components)

}

} catch (error) {

throw new Error('Reverse geocoding failed')

}

}

calculateDistance(

point1: { latitude: number; longitude: number },

point2: { latitude: number; longitude: number }

): number {

const R = 6371 // Earth's radius in kilometers

const dLat = this.toRadians(point2.latitude - point1.latitude)

const dLon = this.toRadians(point2.longitude - point1.longitude)

const a = Math.sin(dLat/2) \* Math.sin(dLat/2) +

Math.cos(this.toRadians(point1.latitude)) \*

Math.cos(this.toRadians(point2.latitude)) \*

Math.sin(dLon/2) \* Math.sin(dLon/2)

const c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1-a))

return R \* c

}

async findNearbyJobs(

userLocation: { latitude: number; longitude: number },

radiusKm: number,

skillCategories?: string[]

): Promise

**9. Security Architecture**

**9.1 Authentication & Authorization**

// middleware/AuthMiddleware.ts

import jwt from 'jsonwebtoken'

import rateLimit from 'express-rate-limit'

class AuthMiddleware {

// JWT token verification middleware

static verifyToken(req: Request, res: Response, next: NextFunction) {

const token = req.headers.authorization?.replace('Bearer ', '')

if (!token) {

return res.status(401).json({ error: 'Access token required' })

}

try {

const decoded = jwt.verify(token, process.env.JWT\_SECRET!) as JWTPayload

req.user = decoded

next()

} catch (error) {

if (error.name === 'TokenExpiredError') {

return res.status(401).json({ error: 'Token expired' })

}

return res.status(401).json({ error: 'Invalid token' })

}

}

// Role-based access control

static requireRole(roles: UserRole[]) {

return (req: Request, res: Response, next: NextFunction) => {

if (!req.user || !roles.includes(req.user.userType)) {

return res.status(403).json({ error: 'Insufficient permissions' })

}

next()

}

}

// Rate limiting by user type

static createRateLimit(windowMs: number, maxRequests: number) {

return rateLimit({

windowMs,

max: maxRequests,

keyGenerator: (req) => {

return req.user?.userId || req.ip

},

message: 'Too many requests, please try again later'

})

}

// API key validation for external integrations

static verifyApiKey(req: Request, res: Response, next: NextFunction) {

const apiKey = req.headers['x-api-key']

if (!apiKey || !this.isValidApiKey(apiKey as string)) {

return res.status(401).json({ error: 'Invalid API key' })

}

next()

}

private static isValidApiKey(apiKey: string): boolean {

const validApiKeys = process.env.VALID\_API\_KEYS?.split(',') || []

return validApiKeys.includes(apiKey)

}

}

**9.2 Data Encryption & Privacy**

// utils/EncryptionUtils.ts

import crypto from 'crypto'

import bcrypt from 'bcrypt'

class EncryptionUtils {

private static readonly ALGORITHM = 'aes-256-gcm'

private static readonly SECRET\_KEY = process.env.ENCRYPTION\_KEY!

// Encrypt sensitive data

static encrypt(text: string): EncryptedData {

const iv = crypto.randomBytes(16)

const cipher = crypto.createCipher(this.ALGORITHM, this.SECRET\_KEY)

cipher.setAAD(Buffer.from('additional-auth-data'))

let encrypted = cipher.update(text, 'utf8', 'hex')

encrypted += cipher.final('hex')

const authTag = cipher.getAuthTag()

return {

encryptedData: encrypted,

iv: iv.toString('hex'),

authTag: authTag.toString('hex')

}

}

// Decrypt sensitive data

static decrypt(encryptedData: EncryptedData): string {

const decipher = crypto.createDecipher(this.ALGORITHM, this.SECRET\_KEY)

decipher.setAAD(Buffer.from('additional-auth-data'))

decipher.setAuthTag(Buffer.from(encryptedData.authTag, 'hex'))

let decrypted = decipher.update(encryptedData.encryptedData, 'hex', 'utf8')

decrypted += decipher.final('utf8')

return decrypted

}

// Hash passwords

static async hashPassword(password: string): Promise<string> {

const saltRounds = 12

return bcrypt.hash(password, saltRounds)

}

// Verify password

static async verifyPassword(password: string, hash: string): Promise<boolean> {

return bcrypt.compare(password, hash)

}

// Generate secure random tokens

static generateSecureToken(length: number = 32): string {

return crypto.randomBytes(length).toString('hex')

}

// Hash sensitive identifiers (phone numbers, emails)

static hashIdentifier(identifier: string): string {

return crypto

.createHash('sha256')

.update(identifier + process.env.HASH\_SALT!)

.digest('hex')

}

}

**9.3 Input Validation & Sanitization**

// utils/ValidationUtils.ts

import Joi from 'joi'

import DOMPurify from 'isomorphic-dompurify'

class ValidationUtils {

// Common validation schemas

static readonly schemas = {

phoneNumber: Joi.string()

.pattern(/^[6-9]\d{9}$/)

.required()

.messages({

'string.pattern.base': 'Please enter a valid Indian mobile number'

}),

email: Joi.string()

.email()

.optional(),

aadhaarNumber: Joi.string()

.pattern(/^\d{12}$/)

.optional(),

panNumber: Joi.string()

.pattern(/^[A-Z]{5}[0-9]{4}[A-Z]$/)

.optional(),

jobTitle: Joi.string()

.min(3)

.max(100)

.required(),

jobDescription: Joi.string()

.min(10)

.max(2000)

.required(),

coordinates: Joi.object({

latitude: Joi.number().min(-90).max(90).required(),

longitude: Joi.number().min(-180).max(180).required()

}).required()

}

// Validate and sanitize user input

static validateUserInput(data: any, schema: Joi.ObjectSchema) {

const { error, value } = schema.validate(data, {

abortEarly: false,

stripUnknown: true

})

if (error) {

const errors = error.details.map(detail => ({

field: detail.path.join('.'),

message: detail.message

}))

throw new ValidationError('Validation failed', errors)

}

return this.sanitizeData(value)

}

// Sanitize HTML content and prevent XSS

static sanitizeData(data: any): any {

if (typeof data === 'string') {

return DOMPurify.sanitize(data.trim())

}

if (Array.isArray(data)) {

return data.map(item => this.sanitizeData(item))

}

if (typeof data === 'object' && data !== null) {

const sanitized: any = {}

for (const key in data) {

sanitized[key] = this.sanitizeData(data[key])

}

return sanitized

}

return data

}

// SQL injection prevention (using parameterized queries)

static sanitizeQuery(query: string): string {

// Remove common SQL injection patterns

const dangerousPatterns = [

/(\s|^)(DROP|DELETE|INSERT|UPDATE|ALTER|CREATE)\s+/gi,

/(\s|^)(UNION|SELECT|FROM|WHERE)\s+/gi,

/--|\#|\/\\*/g,

/'/g

]

let sanitized = query

dangerousPatterns.forEach(pattern => {

sanitized = sanitized.replace(pattern, '')

})

return sanitized.trim()

}

// File upload validation

static validateFileUpload(file: Express.Multer.File) {

const allowedTypes = [

'image/jpeg', 'image/jpg', 'image/png',

'application/pdf', 'audio/mpeg', 'audio/wav'

]

const maxSize = 5 \* 1024 \* 1024 // 5MB

if (!allowedTypes.includes(file.mimetype)) {

throw new Error('File type not allowed')

}

if (file.size > maxSize) {

throw new Error('File size too large')

}

// Check for malicious file signatures

this.validateFileSignature(file.buffer, file.mimetype)

return true

}

private static validateFileSignature(buffer: Buffer, mimeType: string) {

const signatures: { [key: string]: string[] } = {

'image/jpeg': ['FFD8FF'],

'image/png': ['89504E47'],

'application/pdf': ['25504446']

}

const fileSignature = buffer.toString('hex', 0, 4).toUpperCase()

const expectedSignatures = signatures[mimeType]

if (expectedSignatures && !expectedSignatures.some(sig => fileSignature.startsWith(sig))) {

throw new Error('File signature mismatch')

}

}

}

**10. Scalability & Performance**

**10.1 Database Optimization Strategies**

-- Database Indexing Strategy

-- Users table indexes

CREATE INDEX CONCURRENTLY idx\_users\_phone ON users(phone\_number) WHERE deleted\_at IS NULL;

CREATE INDEX CONCURRENTLY idx\_users\_type\_status ON users(user\_type, status) WHERE deleted\_at IS NULL;

-- Job postings indexes for search optimization

CREATE INDEX CONCURRENTLY idx\_jobs\_location\_status ON job\_postings

USING GIST(job\_location) WHERE status = 'active' AND expires\_at > NOW();

CREATE INDEX CONCURRENTLY idx\_jobs\_category\_urgency ON job\_postings(category\_id, urgency)

WHERE status = 'active' AND expires\_at > NOW();

CREATE INDEX CONCURRENTLY idx\_jobs\_posted\_date ON job\_postings(created\_at DESC)

WHERE status = 'active';

-- Job applications indexes

CREATE INDEX CONCURRENTLY idx\_applications\_job\_seeker ON job\_applications(job\_seeker\_id, status);

CREATE INDEX CONCURRENTLY idx\_applications\_job\_employer ON job\_applications(job\_posting\_id, status);

CREATE INDEX CONCURRENTLY idx\_applications\_applied\_date ON job\_applications(applied\_at DESC);

-- Skills and matching indexes

CREATE INDEX CONCURRENTLY idx\_job\_seeker\_skills ON job\_seeker\_skills(job\_seeker\_id, skill\_id);

CREATE INDEX CONCURRENTLY idx\_skills\_category ON skills(category\_id) WHERE is\_active = true;

-- Messages and conversations

CREATE INDEX CONCURRENTLY idx\_messages\_conversation\_date ON messages(conversation\_id, sent\_at DESC);

CREATE INDEX CONCURRENTLY idx\_conversations\_participants ON conversations(employer\_id, job\_seeker\_id);

-- Partitioning for large tables (job applications, messages, activity logs)

CREATE TABLE job\_applications\_y2024m01 PARTITION OF job\_applications

FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');

CREATE TABLE job\_applications\_y2024m02 PARTITION OF job\_applications

FOR VALUES FROM ('2024-02-01') TO ('2024-03-01');

**10.2 Caching Strategy**

// services/cache/CacheService.ts

import Redis from 'ioredis'

class CacheService {

private redis: Redis

private readonly DEFAULT\_TTL = 3600 // 1 hour

constructor() {

this.redis = new Redis({

host: process.env.REDIS\_HOST,

port: parseInt(process.env.REDIS\_PORT!),

password: process.env.REDIS\_PASSWORD,

retryDelayOnFailover: 100,

maxRetriesPerRequest: 3

})

}

// Multi-level caching strategy

async getJobSearchResults(searchQuery: SearchQuery): Promise<JobSearchResult[]> {

const cacheKey = this.generateSearchCacheKey(searchQuery)

// Level 1: Redis cache

const cachedResults = await this.redis.get(cacheKey)

if (cachedResults) {

return JSON.parse(cachedResults)

}

// Level 2: Database query

const results = await this.jobSearchService.searchJobs(searchQuery)

// Cache results with appropriate TTL

const ttl = this.calculateCacheTTL(searchQuery)

await this.redis.setex(cacheKey, ttl, JSON.stringify(results))

return results

}

// Intelligent cache invalidation

async invalidateJobCaches(jobId: string) {

const patterns = [

`job\_search:\*:location:\*`, // Location-based searches

`job\_details:${jobId}`,

`recommended\_jobs:\*`, // Recommended jobs for users

`nearby\_jobs:\*` // Nearby job searches

]

for (const pattern of patterns) {

const keys = await this.redis.keys(pattern)

if (keys.length > 0) {

await this.redis.del(...keys)

}

}

}

// Session-based caching for user preferences

async cacheUserSession(userId: string, sessionData: any) {

const sessionKey = `user\_session:${userId}`

await this.redis.hset(sessionKey, {

preferences: JSON.stringify(sessionData.preferences),

lastLocation: JSON.stringify(sessionData.location),

searchHistory: JSON.stringify(sessionData.searches),

lastActive: Date.now()

})

await this.redis.expire(sessionKey, 86400) // 24 hours

}

// Preemptive caching for popular searches

async warmupCache() {

const popularSearches = await this.getPopularSearchQueries()

for (const query of popularSearches) {

// Pre-load search results

await this.getJobSearchResults(query)

}

}

private generateSearchCacheKey(query: SearchQuery): string {

const keyParts = [

'job\_search',

query.location ? `lat:${query.location.latitude}\_lng:${query.location.longitude}` : 'no\_location',

query.radius ? `r:${query.radius}` : 'r:10',

query.category || 'all\_categories',

query.minSalary ? `min:${query.minSalary}` : '',

query.maxSalary ? `max:${query.maxSalary}` : ''

].filter(Boolean)

return keyParts.join(':')

}

private calculateCacheTTL(query: SearchQuery): number {

// Dynamic TTL based on search specificity

let ttl = this.DEFAULT\_TTL

if (query.location) ttl \*= 0.5 // Location searches change more frequently

if (query.urgency === 'immediate') ttl \*= 0.25 // Urgent jobs change very quickly

if (query.category) ttl \*= 1.5 // Category searches are more stable

return Math.max(300, Math.min(ttl, 7200)) // Between 5 minutes and 2 hours

}

}

**10.3 Load Balancing & Auto-scaling**

# Kubernetes deployment configuration

apiVersion: apps/v1

kind: Deployment

metadata:

name: job-portal-api

spec:

replicas: 3

selector:

matchLabels:

app: job-portal-api

template:

metadata:

labels:

app: job-portal-api

spec:

containers:

- name: api

image: job-portal/api:latest

ports:

- containerPort: 3000

env:

- name: NODE\_ENV

value: "production"

- name: DATABASE\_URL

valueFrom:

secretKeyRef:

name: db-credentials

key: url

resources:

requests:

memory: "256Mi"

cpu: "250m"

limits:

memory: "512Mi"

cpu: "500m"

livenessProbe:

httpGet:

path: /health

port: 3000

initialDelaySeconds: 30

periodSeconds: 10

readinessProbe:

httpGet:

path: /ready

port: 3000

initialDelaySeconds: 5

periodSeconds: 5

---

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: job-portal-api-hpa

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: job-portal-api

minReplicas: 3

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 70

- type: Resource

resource:

name: memory

target:

type: Utilization

averageUtilization: 80

**10.4 CDN & Asset Optimization**

// services/cdn/AssetOptimizationService.ts

class AssetOptimizationService {

private cdnBaseUrl: string = process.env.CDN\_BASE\_URL!

// Generate optimized image URLs with transformations

generateImageUrl(

originalUrl: string,

options: ImageOptions = {}

): string {

const {

width = 400,

height = 400,

quality = 80,

format = 'webp',

fit = 'cover'

} = options

const params = new URLSearchParams({

w: width.toString(),

h: height.toString(),

q: quality.toString(),

f: format,

fit

})

return `${this.cdnBaseUrl}/images/transform?url=${encodeURIComponent(originalUrl)}&${params.toString()}`

}

// Responsive image URLs for different screen sizes

generateResponsiveImageUrls(originalUrl: string): ResponsiveImageUrls {

return {

small: this.generateImageUrl(originalUrl, { width: 200, quality: 70 }),

medium: this.generateImageUrl(originalUrl, { width: 400, quality: 75 }),

large: this.generateImageUrl(originalUrl, { width: 800, quality: 80 }),

xlarge: this.generateImageUrl(originalUrl, { width: 1200, quality: 85 })

}

}

// Optimize document URLs for fast delivery

generateDocumentUrl(documentId: string, options: DocumentOptions = {}): string {

const {

thumbnail = false,

watermark = false

} = options

const params = new URLSearchParams()

if (thumbnail) params.append('thumbnail', 'true')

if (watermark) params.append('watermark', 'true')

return `${this.cdnBaseUrl}/documents/${documentId}?${params.toString()}`

}

// Audio file optimization for voice messages

generateAudioUrl(audioId: string, quality: 'low' | 'medium' | 'high' = 'medium'): string {

const qualityMap = {

low: '32k',

medium: '64k',

high: '128k'

}

return `${this.cdnBaseUrl}/audio/${audioId}?bitrate=${qualityMap[quality]}&format=mp3`

}

}

**11. Deployment Architecture**

**11.1 Container Strategy (Docker)**

# Multi-stage Dockerfile for Node.js API

FROM node:18-alpine AS builder

WORKDIR /app

COPY package\*.json ./

RUN npm ci --only=production && npm cache clean --force

# Production stage

FROM node:18-alpine AS production

# Install security updates and necessary packages

RUN apk update && apk upgrade && \

apk add --no-cache dumb-init curl && \

addgroup -g 1001 -S nodejs && \

adduser -S nextjs -u 1001

WORKDIR /app

COPY --from=builder /app/node\_modules ./node\_modules

COPY --chown=nextjs:nodejs . .

USER nextjs

EXPOSE 3000

# Use dumb-init to handle signals properly

ENTRYPOINT ["dumb-init", "--"]

CMD ["node", "dist/server.js"]

# Health check

HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \

CMD curl -f http://localhost:3000/health || exit 1

# React Native Build Container (for CI/CD)

FROM codemagic/android-flutter:latest

WORKDIR /app

# Install Node.js dependencies

COPY package\*.json ./

RUN npm ci

# Copy source code

COPY . .

# Build Android APK

RUN cd android && ./gradlew assembleRelease

# Build Android App Bundle

RUN cd android && ./gradlew bundleRelease

# Extract built artifacts

RUN mkdir -p /artifacts && \

cp android/app/build/outputs/apk/release/app-release.apk /artifacts/ && \

cp android/app/build/outputs/bundle/release/app-release.aab /artifacts/

**11.2 Kubernetes Configuration**

# Namespace configuration

apiVersion: v1

kind: Namespace

metadata:

name: job-portal

labels:

name: job-portal

---

# ConfigMap for application configuration

apiVersion: v1

kind: ConfigMap

metadata:

name: app-config

namespace: job-portal

data:

NODE\_ENV: "production"

LOG\_LEVEL: "info"

API\_VERSION: "v1"

MAX\_FILE\_SIZE: "5MB"

CACHE\_TTL: "3600"

---

# Secrets for sensitive data

apiVersion: v1

kind: Secret

metadata:

name: app-secrets

namespace: job-portal

type: Opaque

data:

DATABASE\_URL: <base64-encoded-database-url>

JWT\_SECRET: <base64-encoded-jwt-secret>

REDIS\_PASSWORD: <base64-encoded-redis-password>

AWS\_ACCESS\_KEY: <base64-encoded-aws-key>

AWS\_SECRET\_KEY: <base64-encoded-aws-secret>

---

# PostgreSQL StatefulSet

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: postgres

namespace: job-portal

spec:

serviceName: postgres

replicas: 1

selector:

matchLabels:

app: postgres

template:

metadata:

labels:

app: postgres

spec:

containers:

- name: postgres

image: postgres:15-alpine

ports:

- containerPort: 5432

env:

- name: POSTGRES\_DB

value: jobportal

- name: POSTGRES\_USER

value: jobportal\_user

- name: POSTGRES\_PASSWORD

valueFrom:

secretKeyRef:

name: app-secrets

key: DATABASE\_PASSWORD

volumeMounts:

- name: postgres-storage

mountPath: /var/lib/postgresql/data

resources:

requests:

memory: "256Mi"

cpu: "250m"

limits:

memory: "1Gi"

cpu: "500m"

volumeClaimTemplates:

- metadata:

name: postgres-storage

spec:

accessModes: ["ReadWriteOnce"]

resources:

requests:

storage: 20Gi

---

# Redis Deployment

apiVersion: apps/v1

kind: Deployment

metadata:

name: redis

namespace: job-portal

spec:

replicas: 1

selector:

matchLabels:

app: redis

template:

metadata:

labels:

app: redis

spec:

containers:

- name: redis

image: redis:7-alpine

ports:

- containerPort: 6379

command: ["redis-server"]

args: ["--requirepass", "$(REDIS\_PASSWORD)", "--appendonly", "yes"]

env:

- name: REDIS\_PASSWORD

valueFrom:

secretKeyRef:

name: app-secrets

key: REDIS\_PASSWORD

volumeMounts:

- name: redis-storage

mountPath: /data

resources:

requests:

memory: "128Mi"

cpu: "100m"

limits:

memory: "256Mi"

cpu: "200m"

volumes:

- name: redis-storage

persistentVolumeClaim:

claimName: redis-pvc

---

# Elasticsearch Deployment

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: elasticsearch

namespace: job-portal

spec:

serviceName: elasticsearch

replicas: 3

selector:

matchLabels:

app: elasticsearch

template:

metadata:

labels:

app: elasticsearch

spec:

containers:

- name: elasticsearch

image: docker.elastic.co/elasticsearch/elasticsearch:8.5.0

ports:

- containerPort: 9200

- containerPort: 9300

env:

- name: cluster.name

value: "job-portal-cluster"

- name: node.name

valueFrom:

fieldRef:

fieldPath: metadata.name

- name: discovery.seed\_hosts

value: "elasticsearch-0.elasticsearch,elasticsearch-1.elasticsearch,elasticsearch-2.elasticsearch"

- name: cluster.initial\_master\_nodes

value: "elasticsearch-0,elasticsearch-1,elasticsearch-2"

- name: ES\_JAVA\_OPTS

value: "-Xms512m -Xmx512m"

- name: xpack.security.enabled

value: "false"

volumeMounts:

- name: elasticsearch-storage

mountPath: /usr/share/elasticsearch/data

resources:

requests:

memory: "1Gi"

cpu: "500m"

limits:

memory: "2Gi"

cpu: "1000m"

volumeClaimTemplates:

- metadata:

name: elasticsearch-storage

spec:

accessModes: ["ReadWriteOnce"]

resources:

requests:

storage: 50Gi

---

# API Gateway/Load Balancer (NGINX)

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-gateway

namespace: job-portal

spec:

replicas: 2

selector:

matchLabels:

app: nginx-gateway

template:

metadata:

labels:

app: nginx-gateway

spec:

containers:

- name: nginx

image: nginx:1.23-alpine

ports:

- containerPort: 80

- containerPort: 443

volumeMounts:

- name: nginx-config

mountPath: /etc/nginx/nginx.conf

subPath: nginx.conf

- name: ssl-certs

mountPath: /etc/ssl/certs

resources:

requests:

memory: "64Mi"

cpu: "50m"

limits:

memory: "128Mi"

cpu: "100m"

volumes:

- name: nginx-config

configMap:

name: nginx-config

- name: ssl-certs

secret:

secretName: tls-secret

**11.3 CI/CD Pipeline Configuration**

# .github/workflows/deploy.yml

name: Build and Deploy

on:

push:

branches: [main, develop]

pull\_request:

branches: [main]

env:

DOCKER\_REGISTRY: ghcr.io

IMAGE\_NAME: jobportal/api

jobs:

test:

runs-on: ubuntu-latest

services:

postgres:

image: postgres:15

env:

POSTGRES\_PASSWORD: test\_password

POSTGRES\_DB: test\_db

options: >-

--health-cmd pg\_isready

--health-interval 10s

--health-timeout 5s

--health-retries 5

redis:

image: redis:7-alpine

options: >-

--health-cmd "redis-cli ping"

--health-interval 10s

--health-timeout 5s

--health-retries 5

steps:

- uses: actions/checkout@v3

- name: Setup Node.js

uses: actions/setup-node@v3

with:

node-version: '18'

cache: 'npm'

- name: Install dependencies

run: npm ci

- name: Run linting

run: npm run lint

- name: Run type checking

run: npm run type-check

- name: Run unit tests

run: npm run test:unit

env:

DATABASE\_URL: postgres://postgres:test\_password@localhost:5432/test\_db

REDIS\_URL: redis://localhost:6379

- name: Run integration tests

run: npm run test:integration

env:

DATABASE\_URL: postgres://postgres:test\_password@localhost:5432/test\_db

REDIS\_URL: redis://localhost:6379

- name: Upload coverage reports

uses: codecov/codecov-action@v3

with:

file: ./coverage/lcov.info

build-and-push:

needs: test

runs-on: ubuntu-latest

if: github.event\_name == 'push'

steps:

- uses: actions/checkout@v3

- name: Set up Docker Buildx

uses: docker/setup-buildx-action@v2

- name: Log in to Container Registry

uses: docker/login-action@v2

with:

registry: ${{ env.DOCKER\_REGISTRY }}

username: ${{ github.actor }}

password: ${{ secrets.GITHUB\_TOKEN }}

- name: Extract metadata

id: meta

uses: docker/metadata-action@v4

with:

images: ${{ env.DOCKER\_REGISTRY }}/${{ env.IMAGE\_NAME }}

tags: |

type=ref,event=branch

type=ref,event=pr

type=sha,prefix={{branch}}-

- name: Build and push Docker image

uses: docker/build-push-action@v4

with:

context: .

platforms: linux/amd64,linux/arm64

push: true

tags: ${{ steps.meta.outputs.tags }}

labels: ${{ steps.meta.outputs.labels }}

cache-from: type=gha

cache-to: type=gha,mode=max

deploy-staging:

needs: build-and-push

runs-on: ubuntu-latest

if: github.ref == 'refs/heads/develop'

environment: staging

steps:

- uses: actions/checkout@v3

- name: Configure kubectl

uses: azure/k8s-set-context@v1

with:

method: kubeconfig

kubeconfig: ${{ secrets.KUBE\_CONFIG\_STAGING }}

- name: Deploy to staging

run: |

envsubst < k8s/deployment.yaml | kubectl apply -f -

kubectl rollout status deployment/job-portal-api -n job-portal-staging

env:

IMAGE\_TAG: ${{ github.sha }}

ENVIRONMENT: staging

deploy-production:

needs: build-and-push

runs-on: ubuntu-latest

if: github.ref == 'refs/heads/main'

environment: production

steps:

- uses: actions/checkout@v3

- name: Configure kubectl

uses: azure/k8s-set-context@v1

with:

method: kubeconfig

kubeconfig: ${{ secrets.KUBE\_CONFIG\_PRODUCTION }}

- name: Deploy to production

run: |

envsubst < k8s/deployment.yaml | kubectl apply -f -

kubectl rollout status deployment/job-portal-api -n job-portal-production

env:

IMAGE\_TAG: ${{ github.sha }}

ENVIRONMENT: production

build-mobile-app:

needs: test

runs-on: macos-latest

if: github.ref == 'refs/heads/main'

steps:

- uses: actions/checkout@v3

- name: Setup Node.js

uses: actions/setup-node@v3

with:

node-version: '18'

cache: 'npm'

- name: Install dependencies

run: npm ci

- name: Setup Ruby for iOS

uses: ruby/setup-ruby@v1

with:

ruby-version: '3.0'

bundler-cache: true

working-directory: ios

- name: Install iOS dependencies

run: |

cd ios

bundle install

pod install

- name: Build iOS app

run: |

cd ios

xcodebuild -workspace JobPortal.xcworkspace \

-scheme JobPortal \

-configuration Release \

-archivePath JobPortal.xcarchive \

archive

- name: Setup Java for Android

uses: actions/setup-java@v3

with:

distribution: 'zulu'

java-version: '11'

- name: Setup Android SDK

uses: android-actions/setup-android@v2

- name: Build Android APK

run: |

cd android

./gradlew assembleRelease

- name: Upload Android APK

uses: actions/upload-artifact@v3

with:

name: android-apk

path: android/app/build/outputs/apk/release/app-release.apk

**11.4 Infrastructure as Code (Terraform)**

# infrastructure/main.tf

terraform {

required\_version = ">= 1.0"

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 5.0"

}

kubernetes = {

source = "hashicorp/kubernetes"

version = "~> 2.23"

}

}

backend "s3" {

bucket = "jobportal-terraform-state"

key = "infrastructure/terraform.tfstate"

region = "ap-south-1"

}

}

provider "aws" {

region = var.aws\_region

}

# VPC Configuration

module "vpc" {

source = "terraform-aws-modules/vpc/aws"

name = "jobportal-vpc"

cidr = "10.0.0.0/16"

azs = ["${var.aws\_region}a", "${var.aws\_region}b", "${var.aws\_region}c"]

private\_subnets = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]

public\_subnets = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]

enable\_nat\_gateway = true

enable\_vpn\_gateway = true

enable\_dns\_hostnames = true

enable\_dns\_support = true

tags = {

Environment = var.environment

Project = "JobPortal"

}

}

# EKS Cluster

module "eks" {

source = "terraform-aws-modules/eks/aws"

cluster\_name = "jobportal-${var.environment}"

cluster\_version = "1.28"

vpc\_id = module.vpc.vpc\_id

subnet\_ids = module.vpc.private\_subnets

# EKS Managed Node Groups

eks\_managed\_node\_groups = {

main = {

min\_size = 2

max\_size = 10

desired\_size = 3

instance\_types = ["t3.medium"]

k8s\_labels = {

Environment = var.environment

NodeType = "main"

}

}

compute = {

min\_size = 1

max\_size = 5

desired\_size = 2

instance\_types = ["c5.large"]

k8s\_labels = {

Environment = var.environment

NodeType = "compute"

}

taints = [

{

key = "compute-intensive"

value = "true"

effect = "NO\_SCHEDULE"

}

]

}

}

tags = {

Environment = var.environment

Project = "JobPortal"

}

}

# RDS PostgreSQL

resource "aws\_db\_subnet\_group" "main" {

name = "jobportal-${var.environment}"

subnet\_ids = module.vpc.private\_subnets

tags = {

Name = "JobPortal DB subnet group"

Environment = var.environment

}

}

resource "aws\_security\_group" "rds" {

name\_prefix = "jobportal-rds-${var.environment}"

vpc\_id = module.vpc.vpc\_id

ingress {

from\_port = 5432

to\_port = 5432

protocol = "tcp"

cidr\_blocks = [module.vpc.vpc\_cidr\_block]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "JobPortal RDS Security Group"

Environment = var.environment

}

}

resource "aws\_db\_instance" "postgres" {

allocated\_storage = var.db\_allocated\_storage

max\_allocated\_storage = var.db\_max\_allocated\_storage

engine = "postgres"

engine\_version = "15.4"

instance\_class = var.db\_instance\_class

db\_name = "jobportal"

username = "jobportal\_user"

password = var.db\_password

vpc\_security\_group\_ids = [aws\_security\_group.rds.id]

db\_subnet\_group\_name = aws\_db\_subnet\_group.main.name

backup\_retention\_period = 7

backup\_window = "03:00-04:00"

maintenance\_window = "sun:04:00-sun:05:00"

skip\_final\_snapshot = var.environment == "development"

deletion\_protection = var.environment == "production"

performance\_insights\_enabled = true

monitoring\_interval = 60

tags = {

Name = "JobPortal PostgreSQL"

Environment = var.environment

}

}

# ElastiCache Redis

resource "aws\_elasticache\_subnet\_group" "main" {

name = "jobportal-${var.environment}"

subnet\_ids = module.vpc.private\_subnets

}

resource "aws\_security\_group" "redis" {

name\_prefix = "jobportal-redis-${var.environment}"

vpc\_id = module.vpc.vpc\_id

ingress {

from\_port = 6379

to\_port = 6379

protocol = "tcp"

cidr\_blocks = [module.vpc.vpc\_cidr\_block]

}

tags = {

Name = "JobPortal Redis Security Group"

Environment = var.environment

}

}

resource "aws\_elasticache\_replication\_group" "redis" {

replication\_group\_id = "jobportal-${var.environment}"

description = "Redis cluster for JobPortal"

node\_type = var.redis\_node\_type

port = 6379

parameter\_group\_name = "default.redis7"

num\_cache\_clusters = var.redis\_num\_nodes

automatic\_failover\_enabled = var.redis\_num\_nodes > 1

multi\_az\_enabled = var.redis\_num\_nodes > 1

subnet\_group\_name = aws\_elasticache\_subnet\_group.main.name

security\_group\_ids = [aws\_security\_group.redis.id]

at\_rest\_encryption\_enabled = true

transit\_encryption\_enabled = true

auth\_token = var.redis\_auth\_token

log\_delivery\_configuration {

destination = aws\_cloudwatch\_log\_group.redis.name

destination\_type = "cloudwatch-logs"

log\_format = "text"

log\_type = "slow-log"

}

tags = {

Name = "JobPortal Redis"

Environment = var.environment

}

}

# S3 Buckets

resource "aws\_s3\_bucket" "app\_storage" {

bucket = "jobportal-${var.environment}-storage"

tags = {

Name = "JobPortal App Storage"

Environment = var.environment

}

}

resource "aws\_s3\_bucket\_versioning" "app\_storage" {

bucket = aws\_s3\_bucket.app\_storage.id

versioning\_configuration {

status = "Enabled"

}

}

resource "aws\_s3\_bucket\_encryption" "app\_storage" {

bucket = aws\_s3\_bucket.app\_storage.id

server\_side\_encryption\_configuration {

rule {

apply\_server\_side\_encryption\_by\_default {

sse\_algorithm = "AES256"

}

}

}

}

# CloudFront Distribution

resource "aws\_cloudfront\_distribution" "cdn" {

origin {

domain\_name = aws\_s3\_bucket.app\_storage.bucket\_regional\_domain\_name

origin\_id = "S3-${aws\_s3\_bucket.app\_storage.id}"

s3\_origin\_config {

origin\_access\_identity = aws\_cloudfront\_origin\_access\_identity.main.cloudfront\_access\_identity\_path

}

}

enabled = true

default\_root\_object = "index.html"

default\_cache\_behavior {

allowed\_methods = ["DELETE", "GET", "HEAD", "OPTIONS", "PATCH", "POST", "PUT"]

cached\_methods = ["GET", "HEAD"]

target\_origin\_id = "S3-${aws\_s3\_bucket.app\_storage.id}"

compress = true

viewer\_protocol\_policy = "redirect-to-https"

forwarded\_values {

query\_string = false

cookies {

forward = "none"

}

}

min\_ttl = 0

default\_ttl = 3600

max\_ttl = 86400

}

# Cache behavior for images

ordered\_cache\_behavior {

path\_pattern = "/images/\*"

allowed\_methods = ["GET", "HEAD", "OPTIONS"]

cached\_methods = ["GET", "HEAD"]

target\_origin\_id = "S3-${aws\_s3\_bucket.app\_storage.id}"

compress = true

forwarded\_values {

query\_string = true

cookies {

forward = "none"

}

}

min\_ttl = 0

default\_ttl = 86400

max\_ttl = 31536000

viewer\_protocol\_policy = "redirect-to-https"

}

price\_class = "PriceClass\_All"

restrictions {

geo\_restriction {

restriction\_type = "whitelist"

locations = ["IN"] # India only

}

}

viewer\_certificate {

cloudfront\_default\_certificate = true

}

tags = {

Name = "JobPortal CDN"

Environment = var.environment

}

}

# Variables

variable "environment" {

description = "Environment name"

type = string

}

variable "aws\_region" {

description = "AWS region"

type = string

default = "ap-south-1"

}

variable "db\_allocated\_storage" {

description = "Initial allocated storage for RDS"

type = number

default = 100

}

variable "db\_max\_allocated\_storage" {

description = "Maximum allocated storage for RDS"

type = number

default = 1000

}

variable "db\_instance\_class" {

description = "RDS instance class"

type = string

default = "db.t3.medium"

}

variable "db\_password" {

description = "Database password"

type = string

sensitive = true

}

variable "redis\_node\_type" {

description = "Redis node type"

type = string

default = "cache.t3.medium"

}

variable "redis\_num\_nodes" {

description = "Number of Redis nodes"

type = number

default = 2

}

variable "redis\_auth\_token" {

description = "Redis authentication token"

type = string

sensitive = true

}

# Outputs

output "eks\_cluster\_endpoint" {

description = "EKS cluster endpoint"

value = module.eks.cluster\_endpoint

}

output "rds\_endpoint" {

description = "RDS instance endpoint"

value = aws\_db\_instance.postgres.endpoint

}

output "redis\_endpoint" {

description = "Redis cluster endpoint"

value = aws\_elasticache\_replication\_group.redis.configuration\_endpoint\_address

}

output "cloudfront\_distribution\_domain" {

description = "CloudFront distribution domain name"

value = aws\_cloudfront\_distribution.cdn.domain\_name

}

**12. Monitoring & Analytics**

**12.1 Application Monitoring**

# monitoring/prometheus-config.yml

apiVersion: v1

kind: ConfigMap

metadata:

name: prometheus-config

namespace: monitoring

data:

prometheus.yml: |

global:

scrape\_interval: 15s

evaluation\_interval: 15s

rule\_files:

- "alert\_rules.yml"

alerting:

alertmanagers:

- static\_configs:

- targets:

- alertmanager:9093

scrape\_configs:

# Kubernetes API server

- job\_name: 'kubernetes-apiservers'

kubernetes\_sd\_configs:

- role: endpoints

scheme: https

tls\_config:

ca\_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt

bearer\_token\_file: /var/run/secrets/kubernetes.io/serviceaccount/token

relabel\_configs:

- source\_labels: [\_\_meta\_kubernetes\_namespace, \_\_meta\_kubernetes\_service\_name, \_\_meta\_kubernetes\_endpoint\_port\_name]

action: keep

regex: default;kubernetes;https

# Job Portal API metrics

- job\_name: 'job-portal-api'

static\_configs:

- targets: ['job-portal-api:3000']

metrics\_path: /metrics

scrape\_interval: 10s

# PostgreSQL metrics

- job\_name: 'postgres'

static\_configs:

- targets: ['postgres-exporter:9187']

# Redis metrics

- job\_name: 'redis'

static\_configs:

- targets: ['redis-exporter:9121']

# Node metrics

- job\_name: 'node-exporter'

kubernetes\_sd\_configs:

- role: node

relabel\_configs:

- action: labelmap

regex: \_\_meta\_kubernetes\_node\_label\_(.+)

// services/monitoring/MetricsService.ts

import prometheus from 'prom-client'

class MetricsService {

private static instance: MetricsService

// Custom metrics

private httpRequestDuration = new prometheus.Histogram({

name: 'http\_request\_duration\_seconds',

help: 'HTTP request duration in seconds',

labelNames: ['method', 'route', 'status\_code'],

buckets: [0.1, 0.3, 0.5, 0.7, 1, 3, 5, 7, 10]

})

private httpRequestTotal = new prometheus.Counter({

name: 'http\_requests\_total',

help: 'Total number of HTTP requests',

labelNames: ['method', 'route', 'status\_code']

})

private jobApplicationsTotal = new prometheus.Counter({

name: 'job\_applications\_total',

help: 'Total number of job applications',

labelNames: ['job\_category', 'user\_location']

})

private activeJobsGauge = new prometheus.Gauge({

name: 'active\_jobs\_total',

help: 'Number of currently active jobs',

labelNames: ['category', 'urgency']

})

private userRegistrationsTotal = new prometheus.Counter({

name: 'user\_registrations\_total',

help: 'Total number of user registrations',

labelNames: ['user\_type', 'registration\_source']

})

private databaseConnectionPool = new prometheus.Gauge({

name: 'database\_connection\_pool\_size',

help: 'Current database connection pool size',

labelNames: ['pool\_name', 'state']

})

private redisOperations = new prometheus.Counter({

name: 'redis\_operations\_total',

help: 'Total Redis operations',

labelNames: ['operation', 'status']

})

constructor() {

// Initialize default metrics collection

prometheus.collectDefaultMetrics({

prefix: 'jobportal\_',

gcDurationBuckets: [0.001, 0.01, 0.1, 1, 2, 5]

})

}

static getInstance(): MetricsService {

if (!MetricsService.instance) {

MetricsService.instance = new MetricsService()

}

return MetricsService.instance

}

// HTTP request metrics

recordHttpRequest(method: string, route: string, statusCode: number, duration: number) {

this.httpRequestDuration

.labels(method, route, statusCode.toString())

.observe(duration)

this.httpRequestTotal

.labels(method, route, statusCode.toString())

.inc()

}

// Business metrics

recordJobApplication(jobCategory: string, userLocation: string) {

this.jobApplicationsTotal

.labels(jobCategory, userLocation)

.inc()

}

updateActiveJobs(category: string, urgency: string, count: number) {

this.activeJobsGauge

.labels(category, urgency)

.set(count)

}

recordUserRegistration(userType: string, source: string) {

this.userRegistrationsTotal

.labels(userType, source)

.inc()

}

// Infrastructure metrics

updateDatabaseConnectionPool(poolName: string, active: number, idle: number) {

this.databaseConnectionPool

.labels(poolName, 'active')

.set(active)

this.databaseConnectionPool

.labels(poolName, 'idle')

.set(idle)

}

recordRedisOperation(operation: string, success: boolean) {

this.redisOperations

.labels(operation, success ? 'success' : 'error')

.inc()

}

// Middleware for Express.js

createHttpMetricsMiddleware() {

return (req: any, res: any, next: any) => {

const start = Date.now()

res.on('finish', () => {

const duration = (Date.now() - start) / 1000

this.recordHttpRequest(

req.method,

req.route?.path || req.path,

res.statusCode,

duration

)

})

next()

}

}

// Get metrics for Prometheus scraping

async getMetrics(): Promise<string> {

return prometheus.register.metrics()

}

}

export default MetricsService.getInstance()

**12.2 Alert Configuration**

# monitoring/alert-rules.yml

groups:

- name: job-portal-alerts

rules:

# High error rate

- alert: HighErrorRate

expr: rate(http\_requests\_total{status\_code=~"5.."}[5m]) > 0.1

for: 2m

labels:

severity: critical

annotations:

summary: "High error rate detected"

description: "Error rate is {{ $value }} errors per second"

# High response time

- alert: HighResponseTime

expr: histogram\_quantile(0.95, rate(http\_request\_duration\_seconds\_bucket[5m])) > 2

for: 5m

labels:

severity: warning

annotations:

summary: "High response time detected"

description: "95th percentile response time is {{ $value }} seconds"

# Database connection issues

- alert: DatabaseConnectionPoolLow

expr: database\_connection\_pool\_size{state="active"} / database\_connection\_pool\_size{state="total"} > 0.8

for: 3m

labels:

severity: warning

annotations:

summary: "Database connection pool utilization high"

description: "Connection pool is {{ $value | humanizePercentage }} utilized"

# Redis connection issues

- alert: RedisHighErrorRate

expr: rate(redis\_operations\_total{status="error"}[5m]) > 0.05

for: 2m

labels:

severity: critical

annotations:

summary: "Redis error rate high"

description: "Redis error rate is {{ $value }} errors per second"

**12.3 Logging Configuration**

```yaml

# logging/fluentd-config.yml

apiVersion: v1

kind: ConfigMap

metadata:

name: fluentd-config

namespace: kube-system

data:

fluent.conf: |

<source>

@type tail

path /var/log/containers/\*job-portal\*.log

pos\_file /var/log/fluentd-containers.log.pos

tag kubernetes.\*

format json

time\_key timestamp

time\_format %Y-%m-%dT%H:%M:%S.%NZ

</source>

<filter kubernetes.\*\*>

@type kubernetes\_metadata

</filter>

# Parse application logs

<filter kubernetes.var.log.containers.\*\*job-portal\*\*.log>

@type parser

key\_name log

reserve\_data true

<parse>

@type json

time\_key timestamp

time\_format %Y-%m-%dT%H:%M:%S.%NZ

</parse>

</filter>

# Add custom fields

<filter kubernetes.var.log.containers.\*\*job-portal\*\*.log>

@type record\_transformer

<record>

application job-portal

environment ${ENVIRONMENT}

cluster\_name ${CLUSTER\_NAME}

</record>

</filter>

# Route to different outputs based on log level

<match kubernetes.var.log.containers.\*\*job-portal\*\*.log>

@type copy

# Send errors to Slack

<store>

@type slack

webhook\_url "#{ENV['SLACK\_WEBHOOK\_URL']}"

channel alerts

username fluentd

icon\_emoji :warning:

title Error Alert

message %s

message\_keys level,message,error,stack

<format>

@type json

</format>

<buffer>

flush\_interval 10s

</buffer>

<filter>

level ERROR

</filter>

</store>

# Send all logs to Elasticsearch

<store>

@type elasticsearch

host "#{ENV['ELASTICSEARCH\_HOST']}"

port "#{ENV['ELASTICSEARCH\_PORT']}"

index\_name job-portal-logs

type\_name \_doc

logstash\_format true

logstash\_prefix job-portal

<buffer>

@type file

path /var/log/fluentd-buffers/kubernetes.system.buffer

flush\_mode interval

retry\_type exponential\_backoff

flush\_thread\_count 2

flush\_interval 5s

retry\_forever

retry\_max\_interval 30

chunk\_limit\_size 2M

queue\_limit\_length 8

overflow\_action block

</buffer>

</store>

</match>

// utils/Logger.ts

import winston from 'winston'

import DailyRotateFile from 'winston-daily-rotate-file'

class Logger {

private logger: winston.Logger

constructor() {

const logFormat = winston.format.combine(

winston.format.timestamp(),

winston.format.errors({ stack: true }),

winston.format.json(),

winston.format.printf(({ timestamp, level, message, ...meta }) => {

return JSON.stringify({

timestamp,

level,

message,

service: 'job-portal-api',

environment: process.env.NODE\_ENV,

...meta

})

})

)

const transports: winston.transport[] = [

// Console output for development

new winston.transports.Console({

format: winston.format.combine(

winston.format.colorize(),

winston.format.simple()

)

}),

// File rotation for production

new DailyRotateFile({

filename: 'logs/application-%DATE%.log',

datePattern: 'YYYY-MM-DD',

zippedArchive: true,

maxSize: '20m',

maxFiles: '14d',

format: logFormat

}),

// Error-specific log file

new DailyRotateFile({

filename: 'logs/error-%DATE%.log',

datePattern: 'YYYY-MM-DD',

level: 'error',

zippedArchive: true,

maxSize: '20m',

maxFiles: '30d',

format: logFormat

})

]

this.logger = winston.createLogger({

level: process.env.LOG\_LEVEL || 'info',

format: logFormat,

transports,

exceptionHandlers: [

new winston.transports.File({ filename: 'logs/exceptions.log' })

],

rejectionHandlers: [

new winston.transports.File({ filename: 'logs/rejections.log' })

]

})

}

// Structured logging methods

logUserAction(userId: string, action: string, metadata: any = {}) {

this.logger.info('User action', {

userId,

action,

category: 'user\_action',

...metadata

})

}

logJobAction(jobId: string, action: string, userId: string, metadata: any = {}) {

this.logger.info('Job action', {

jobId,

userId,

action,

category: 'job\_action',

...metadata

})

}

logAPIRequest(req: any, res: any, responseTime: number) {

this.logger.info('API Request', {

method: req.method,

url: req.url,

statusCode: res.statusCode,

responseTime,

userAgent: req.get('User-Agent'),

ip: req.ip,

userId: req.user?.userId,

category: 'api\_request'

})

}

logError(error: Error, context: any = {}) {

this.logger.error('Application error', {

error: error.message,

stack: error.stack,

category: 'application\_error',

...context

})

}

logSecurityEvent(event: string, userId?: string, metadata: any = {}) {

this.logger.warn('Security event', {

event,

userId,

category: 'security',

...metadata

})

}

logPerformanceMetric(metric: string, value: number, unit: string, metadata: any = {}) {

this.logger.info('Performance metric', {

metric,

value,

unit,

category: 'performance',

...metadata

})

}

// Standard logging methods

info(message: string, meta?: any) {

this.logger.info(message, meta)

}

warn(message: string, meta?: any) {

this.logger.warn(message, meta)

}

error(message: string, meta?: any) {

this.logger.error(message, meta)

}

debug(message: string, meta?: any) {

this.logger.debug(message, meta)

}

}

export default new Logger()

**12.4 Business Analytics**

// services/analytics/AnalyticsService.ts

import { Client } from '@elastic/elasticsearch'

import mixpanel from 'mixpanel'

class AnalyticsService {

private elasticsearch: Client

private mixpanel: typeof mixpanel

constructor() {

this.elasticsearch = new Client({

node: process.env.ELASTICSEARCH\_URL

})

this.mixpanel = mixpanel.init(process.env.MIXPANEL\_TOKEN!)

}

// Track user events

async trackUserEvent(userId: string, event: string, properties: any = {}) {

const eventData = {

user\_id: userId,

timestamp: new Date().toISOString(),

event,

...properties

}

// Send to Mixpanel for real-time analytics

this.mixpanel.track(event, {

distinct\_id: userId,

...properties

})

// Store in Elasticsearch for detailed analysis

await this.elasticsearch.index({

index: 'user-events',

body: eventData

})

}

// Track job-related events

async trackJobEvent(jobId: string, event: string, userId?: string, properties: any = {}) {

const eventData = {

job\_id: jobId,

user\_id: userId,

timestamp: new Date().toISOString(),

event,

...properties

}

await this.elasticsearch.index({

index: 'job-events',

body: eventData

})

if (userId) {

this.mixpanel.track(event, {

distinct\_id: userId,

job\_id: jobId,

...properties

})

}

}

// Generate business insights

async generateJobMatchingInsights(timeRange: string = '7d'): Promise<JobMatchingInsights> {

const query = {

index: 'job-events',

body: {

query: {

bool: {

must: [

{ term: { event: 'job\_application\_submitted' } },

{ range: { timestamp: { gte: `now-${timeRange}` } } }

]

}

},

aggs: {

by\_category: {

terms: { field: 'job\_category', size: 10 }

},

by\_location: {

terms: { field: 'job\_location', size: 20 }

},

by\_salary\_range: {

range: {

field: 'salary\_amount',

ranges: [

{ to: 500 },

{ from: 500, to: 1000 },

{ from: 1000, to: 2000 },

{ from: 2000 }

]

}

},

success\_rate: {

filters: {

filters: {

applications: { term: { event: 'job\_application\_submitted' } },

hires: { term: { event: 'job\_application\_accepted' } }

}

}

}

}

}

}

const response = await this.elasticsearch.search(query)

return this.processJobMatchingData(response.body.aggregations)

}

// User engagement analytics

async analyzeUserEngagement(userId: string): Promise<UserEngagementAnalysis> {

const thirtyDaysAgo = new Date()

thirtyDaysAgo.setDate(thirtyDaysAgo.getDate() - 30)

const query = {

index: 'user-events',

body: {

query: {

bool: {

must: [

{ term: { user\_id: userId } },

{ range: { timestamp: { gte: thirtyDaysAgo.toISOString() } } }

]

}

},

aggs: {

events\_by\_day: {

date\_histogram: {

field: 'timestamp',

calendar\_interval: '1d'

}

},

event\_types: {

terms: { field: 'event', size: 20 }

},

session\_duration: {

scripted\_metric: {

init\_script: 'state.events = []',

map\_script: 'state.events.add(doc.timestamp.value)',

combine\_script: 'return state.events',

reduce\_script: `

List allEvents = [];

for (state in states) {

allEvents.addAll(state);

}

allEvents.sort();

long totalDuration = 0;

int sessions = 0;

long lastEvent = 0;

for (event in allEvents) {

if (lastEvent == 0 || event - lastEvent > 30 \* 60 \* 1000) {

sessions++;

lastEvent = event;

} else {

totalDuration += event - lastEvent;

lastEvent = event;

}

}

return ['total\_duration': totalDuration, 'sessions': sessions];

`

}

}

}

}

}

const response = await this.elasticsearch.search(query)

return this.processEngagementData(response.body.aggregations)

}

// Platform performance analytics

async generatePlatformMetrics(timeRange: string = '24h'): Promise<PlatformMetrics> {

const queries = await Promise.all([

// Total users

this.elasticsearch.count({

index: 'user-events',

body: {

query: {

range: { timestamp: { gte: `now-${timeRange}` } }

}

}

}),

// Job postings

this.elasticsearch.count({

index: 'job-events',

body: {

query: {

bool: {

must: [

{ term: { event: 'job\_posted' } },

{ range: { timestamp: { gte: `now-${timeRange}` } } }

]

}

}

}

}),

// Applications

this.elasticsearch.count({

index: 'job-events',

body: {

query: {

bool: {

must: [

{ term: { event: 'job\_application\_submitted' } },

{ range: { timestamp: { gte: `now-${timeRange}` } } }

]

}

}

}

}),

// Successful hires

this.elasticsearch.count({

index: 'job-events',

body: {

query: {

bool: {

must: [

{ term: { event: 'job\_application\_accepted' } },

{ range: { timestamp: { gte: `now-${timeRange}` } } }

]

}

}

}

})

])

return {

timeRange,

activeUsers: queries[0].body.count,

jobPostings: queries[1].body.count,

applications: queries[2].body.count,

successfulHires: queries[3].body.count,

conversionRate: queries[3].body.count / queries[2].body.count,

timestamp: new Date().toISOString()

}

}

private processJobMatchingData(aggregations: any): JobMatchingInsights {

return {

topCategories: aggregations.by\_category.buckets.map((bucket: any) => ({

category: bucket.key,

applications: bucket.doc\_count

})),

topLocations: aggregations.by\_location.buckets.map((bucket: any) => ({

location: bucket.key,

applications: bucket.doc\_count

})),

salaryDistribution: aggregations.by\_salary\_range.buckets.map((bucket: any) => ({

range: bucket.key,

applications: bucket.doc\_count

})),

overallSuccessRate: aggregations.success\_rate.buckets.hires.doc\_count /

aggregations.success\_rate.buckets.applications.doc\_count

}

}

private processEngagementData(aggregations: any): UserEngagementAnalysis {

const sessionData = aggregations.session\_duration.value

return {

dailyActivity: aggregations.events\_by\_day.buckets.map((bucket: any) => ({

date: bucket.key\_as\_string,

events: bucket.doc\_count

})),

eventBreakdown: aggregations.event\_types.buckets.map((bucket: any) => ({

event: bucket.key,

count: bucket.doc\_count

})),

avgSessionDuration: sessionData.total\_duration / sessionData.sessions,

totalSessions: sessionData.sessions

}

}

}

// Types for analytics data

interface JobMatchingInsights {

topCategories: Array<{ category: string; applications: number }>

topLocations: Array<{ location: string; applications: number }>

salaryDistribution: Array<{ range: string; applications: number }>

overallSuccessRate: number

}

interface UserEngagementAnalysis {

dailyActivity: Array<{ date: string; events: number }>

eventBreakdown: Array<{ event: string; count: number }>

avgSessionDuration: number

totalSessions: number

}

interface PlatformMetrics {

timeRange: string

activeUsers: number

jobPostings: number

applications: number

successfulHires: number

conversionRate: number

timestamp: string

}

export default new AnalyticsService()

**async findNearbyJobs(**

userLocation: { latitude: number; longitude: number },

radiusKm: number,

skillCategories?: string[]

): Promise<NearbyJob[]> { // This would integrate with your Elasticsearch service // to find jobs within the specified radius const jobs = await this.jobSearchService.searchByLocation( userLocation, radiusKm, skillCategories )

return jobs.map(job => ({

...job,

distance: this.calculateDistance(userLocation, job.location),

travelTime: this.estimateTravelTime(userLocation, job.location)

}))

}

private async estimateTravelTime( origin: { latitude: number; longitude: number }, destination: { latitude: number; longitude: number } ): Promise<number> { try { const response = await axios.get( 'https://maps.googleapis.com/maps/api/distancematrix/json', { params: { origins: ${origin.latitude},${origin.longitude}, destinations: ${destination.latitude},${destination.longitude}, mode: 'transit', // Public transport for most users key: this.googleMapsApiKey } } )

const element = response.data.rows[0].elements[0]

return element.duration ? element.duration.value / 60 : 0 // Convert to minutes

} catch (error) {

return 0

}

}

private extractCity(addressComponents: any[]): string { const cityComponent = addressComponents.find(component => component.types.includes('locality') || component.types.includes('administrative\_area\_level\_2') ) return cityComponent?.long\_name || '' }

private extractState(addressComponents: any[]): string { const stateComponent = addressComponents.find(component => component.types.includes('administrative\_area\_level\_1') ) return stateComponent?.long\_name || '' }

private extractPincode(addressComponents: any[]): string { const pincodeComponent = addressComponents.find(component => component.types.includes('postal\_code') ) return pincodeComponent?.long\_name || '' }

private toRadians(degrees: number): number { return degrees \* (Math.PI / 180) } }

### 8.4 Document Verification Integration

```typescript

// services/integrations/DocumentVerificationService.ts

import axios from 'axios'

class DocumentVerificationService {

private aadhaarApiKey: string

private panApiKey: string

constructor() {

this.aadhaarApiKey = process.env.AADHAAR\_API\_KEY!

this.panApiKey = process.env.PAN\_API\_KEY!

}

async verifyAadhaarNumber(aadhaarNumber: string, name: string): Promise<VerificationResult> {

try {

const response = await axios.post(

'https://api.aadhaarkyc.io/api/v1/aadhaar-verification',

{

aadhaar\_number: aadhaarNumber,

name: name

},

{

headers: {

'Authorization': `Bearer ${this.aadhaarApiKey}`,

'Content-Type': 'application/json'

}

}

)

return {

isValid: response.data.status === 'valid',

matchScore: response.data.match\_score || 0,

details: {

name: response.data.name,

dateOfBirth: response.data.dob,

gender: response.data.gender,

address: response.data.address

},

verificationId: response.data.verification\_id

}

} catch (error) {

throw new Error('Aadhaar verification failed')

}

}

async verifyPANNumber(panNumber: string, name: string): Promise<VerificationResult> {

try {

const response = await axios.post(

'https://api.panverification.io/api/v1/pan-verification',

{

pan\_number: panNumber,

name: name

},

{

headers: {

'Authorization': `Bearer ${this.panApiKey}`,

'Content-Type': 'application/json'

}

}

)

return {

isValid: response.data.pan\_status === 'VALID',

matchScore: response.data.name\_match\_score || 0,

details: {

name: response.data.registered\_name,

category: response.data.category,

aadhaarLinked: response.data.aadhaar\_seeding\_status === 'Y'

},

verificationId: response.data.request\_id

}

} catch (error) {

throw new Error('PAN verification failed')

}

}

async extractDataFromDocument(

documentBuffer: Buffer,

documentType: 'aadhaar' | 'pan' | 'driving\_license'

): Promise<ExtractedData> {

// OCR processing using Tesseract or cloud OCR service

const ocrText = await this.ocrService.extractText(documentBuffer)

switch (documentType) {

case 'aadhaar':

return this.parseAadhaarData(ocrText)

case 'pan':

return this.parsePANData(ocrText)

case 'driving\_license':

return this.parseDLData(ocrText)

default:

throw new Error('Unsupported document type')

}

}

private parseAadhaarData(text: string): ExtractedData {

const aadhaarRegex = /\d{4}\s\d{4}\s\d{4}/

const nameRegex = /Name[:\s]+([A-Za-z\s]+)/i

const dobRegex = /DOB[:\s]+(\d{2}\/\d{2}\/\d{4})/i

const aadhaarMatch = text.match(aadhaarRegex)

const nameMatch = text.match(nameRegex)

const dobMatch = text.match(dobRegex)

return {

documentNumber: aadhaarMatch ? aadhaarMatch[0].replace(/\s/g, '') : '',

name: nameMatch ? nameMatch[1].trim() : '',

dateOfBirth: dobMatch ? dobMatch[1] : '',

extractionConfidence: this.calculateConfidence(text, 'aadhaar')

}

}

private parsePANData(text: string): ExtractedData {

const panRegex = /[A-Z]{5}[0-9]{4}[A-Z]/

const nameRegex = /Name[:\s]+([A-Za-z\s]+)/i

const panMatch = text.match(panRegex)

const nameMatch = text.match(nameRegex)

return {

documentNumber: panMatch ? panMatch[0] : '',

name: nameMatch ? nameMatch[1].trim() : '',

extractionConfidence: this.calculateConfidence(text, 'pan')

}

}

private calculateConfidence(text: string, documentType: string): number {

// Simple confidence calculation based on expected patterns

let confidence = 0

if (documentType === 'aadhaar') {

if (text.includes('Government of India')) confidence += 0.3

if (/\d{4}\s\d{4}\s\d{4}/.test(text)) confidence += 0.4

if (/DOB/.test(text)) confidence += 0.3

} else if (documentType === 'pan') {

if (text.includes('INCOME TAX')) confidence += 0.3

if (/[A-Z]{5}[0-9]{4}[A-Z]/.test(text)) confidence += 0.5

if (text.includes('Permanent Account Number')) confidence += 0.2

}

return Math.min(confidence, 1.0)

}

}

**13. Data Flow Diagrams**

**13.1 User Registration Flow**

[Mobile App]

↓ (Phone Number + OTP Request)

[API Gateway] → [Auth Service] → [SMS Service] → [User's Phone]

↓ (OTP Verification)

[Auth Service] → [Database] (Create/Update User)

↓ (JWT Token)

[Mobile App] ← [API Gateway] ← [Auth Service]

↓ (Profile Setup)

[Profile Service] → [Database] → [File Storage] (Profile Photo)

**13.2 Job Search & Application Flow**

[Mobile App]

↓ (Search Query + Location)

[API Gateway] → [Search Service] → [Elasticsearch] (Job Matching)

↓ (Filter by Location/Skills)

[Location Service] → [Maps API] (Distance Calculation)

↓ (Ranked Results)

[Cache Service] ← [Search Service] ← [Elasticsearch]

↓ (Job Details)

[Mobile App] ← [API Gateway] ← [Search Service]

↓ (Apply for Job)

[Application Service] → [Database] (Create Application)

↓ (Notification)

[Notification Service] → [Push Notification] → [Employer's Device]

→ [SMS Service] → [Employer's Phone]

**13.3 Job Posting & Matching Flow**

[Employer Portal]

↓ (Job Details)

[API Gateway] → [Job Service] → [Database] (Store Job)

↓ (Index Job)

[Job Service] → [Elasticsearch] (Job Indexing)

↓ (Find Matching Workers)

[Matching Service] → [Elasticsearch] (Worker Search)

↓ (Matched Workers)

[Notification Service] → [Push/SMS] → [Worker Devices]

↓ (Analytics)

[Analytics Service] → [Analytics Database]

**13.4 Payment Processing Flow**

[Mobile App/Web Portal]

↓ (Payment Initiation)

[API Gateway] → [Payment Service] → [Payment Gateway] (Razorpay/UPI)

↓ (Payment Verification)

[Payment Service] → [Database] (Update Transaction)

↓ (Commission Calculation)

[Payment Service] → [Accounting Service]

↓ (Worker Payment)

[Payout Service] → [Bank API] (Transfer to Worker)

↓ (Confirmation)

[Notification Service] → [Both Parties]