Diligince.ai Final Complete Architecture

1. Executive Summary

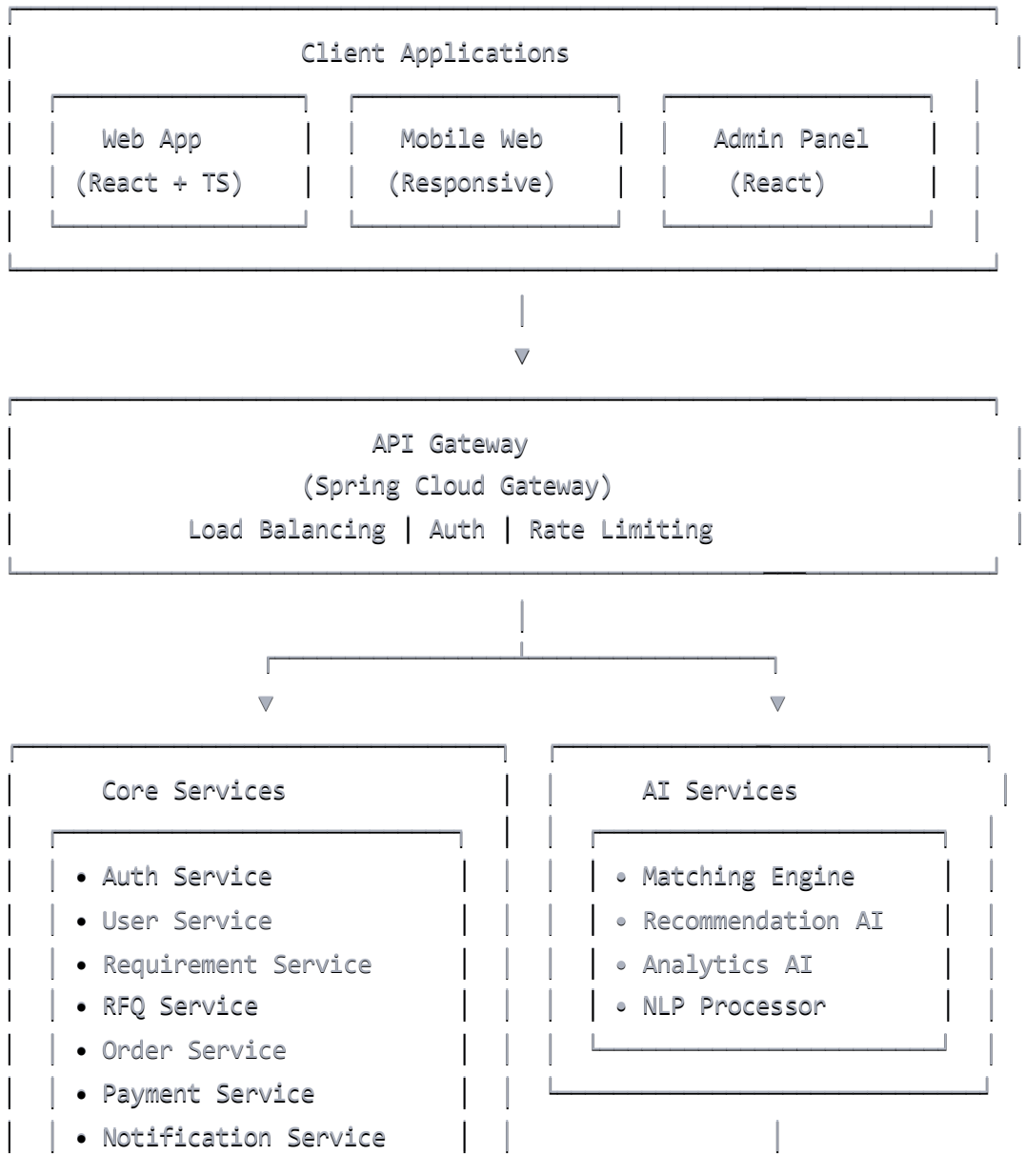
Diligince.ai is a vertical AI-powered B2B SaaS platform that connects Industries with:

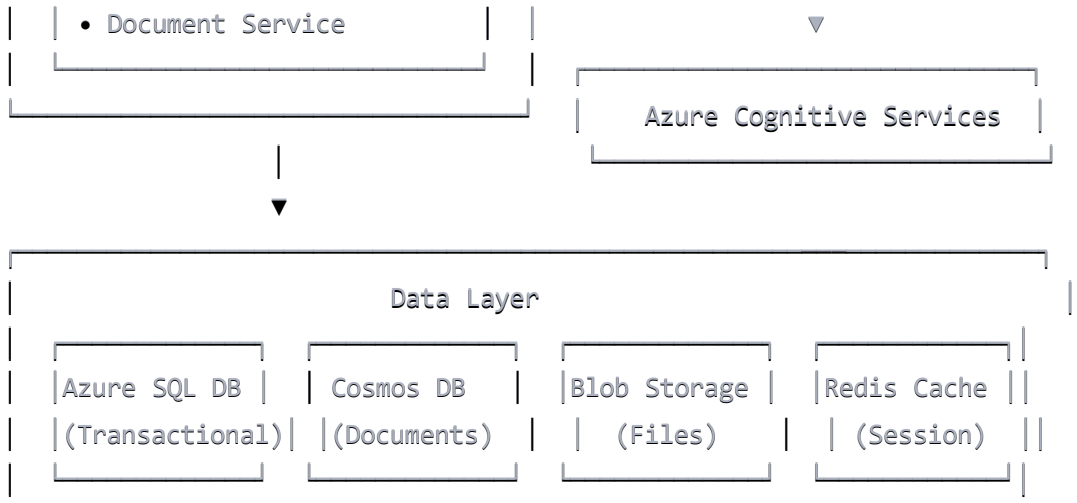
- **Expert Professionals** (Technical Specialists)
- **Service Vendors** (EPCs, Contractors, Consultancies)
- **Product Vendors** (OEMs, Spare Part Suppliers, Material Suppliers)
- **Industrial Logistics Providers**

Core Workflow

Industry posts requirement → AI analyzes & matches → Stakeholders receive RFQs → Quotes comparison
→ PO generation → Project execution → Payment

2. System Architecture Overview




• Document Service
Azure Cognitive Services
Data Layer
Azure SQL DB
(Transactional)
Cosmos DB
(Documents)
Blob Storage
(Files)
Redis Cache
(Session)

3. Frontend Architecture (React + TypeScript + Tailwind CSS)

3.1 Complete Directory Structure

```
diligince-frontend/
├── public/
│   ├── assets/
│   │   ├── images/
│   │   ├── icons/
│   │   └── fonts/
│   ├── favicon.ico
│   └── robots.txt
│
├── src/
│   ├── app/                                # Application core
│   │   ├── App.tsx
│   │   ├── App.test.tsx
│   │   └── routes/
│   │       ├── index.tsx
│   │       ├── PrivateRoute.tsx
│   │       ├── RoleBasedRoute.tsx
│   │       └── routeConfig.ts
│   │
│   ├── features/                           # Feature modules
│   │   ├── auth/
│   │   │   ├── components/
│   │   │   │   ├── LoginForm/
│   │   │   │   ├── RegisterFlow/
│   │   │   │   │   ├── RoleSelection.tsx
│   │   │   │   │   ├── IndustryRegistration.tsx
│   │   │   │   │   ├── ExpertRegistration.tsx
│   │   │   │   │   ├── VendorRegistration.tsx
│   │   │   │   │   └── LogisticsRegistration.tsx
│   │   │   │   └── PasswordReset/
│   │   │   ├── hooks/
│   │   │   │   ├── useAuth.ts
│   │   │   │   └── usePermissions.ts
│   │   │   ├── services/
│   │   │   │   └── auth.service.ts
│   │   │   ├── store/
│   │   │   │   └── authSlice.ts
│   │   │   ├── types/
│   │   │   │   └── auth.types.ts
│   │   │   └── pages/
│   │   │       │   ‖   ‖   |
```

```
│   │   │       ├── Login.tsx
│   │   │       ├── Register.tsx
│   │   │       └── ForgotPassword.tsx
│   │   ├── dashboard/
│   │   │   ├── industry/
```

```
│   │   │   │   ├── components/
│   │   │   │   │   ├── RequirementsSummary.tsx
│   │   │   │   │   ├── ActiveRFQs.tsx
│   │   │   │   │   ├── PendingOrders.tsx
│   │   │   │   │   └── SpendAnalytics.tsx
│   │   │   │   ├── pages/
│   │   │   │   │   └── IndustryDashboard.tsx
│   │   │   │   └── hooks/
│   │   │   ├── expert/
│   │   │   │   ├── components/
│   │   │   │   │   ├── MatchedRequirements.tsx
│   │   │   │   │   ├── ActiveProjects.tsx
│   │   │   │   │   └── EarningsOverview.tsx
│   │   │   │   └── pages/
│   │   │   │       └── ExpertDashboard.tsx
│   │   │   ├── vendor/
│   │   │   │   ├── components/
│   │   │   │   │   ├── RFQInvitations.tsx
│   │   │   │   │   ├── QuoteStatus.tsx
│   │   │   │   │   └── OrderTracking.tsx
│   │   │   │   └── pages/
│   │   │   │       └── VendorDashboard.tsx
│   │   │   └── logistics/
│   │   │       ├── components/
│   │   │       │   ├── TransportRequests.tsx
│   │   │       │   ├── FleetUtilization.tsx
│   │   │       │   └── RouteOptimization.tsx
│   │   │       └── pages/
│   │   │           └── LogisticsDashboard.tsx
│   │   │
│   │   ├── requirements/
│   │   │   ├── components/
│   │   │   │   ├── CreateRequirement/
│   │   │   │   │   ├── RequirementWizard.tsx
│   │   │   │   │   ├── StepIndicator.tsx
│   │   │   │   │   └── steps/
│   │   │   │   │       ├── CategorySelection.tsx
│   │   │   │   │       ├── BasicDetails.tsx
│   │   │   │   │       ├── TechnicalSpecs.tsx
│   │   │   │   │       ├── BudgetTimeline.tsx
│   │   │   │   │       ├── LocationDetails.tsx
│   │   │   │   │       ├── DocumentUpload.tsx
│   │   │   │   │       └── ReviewPublish.tsx
│   │   │   │   ├── RequirementList/
│   │   │   │   │   ├── RequirementCard.tsx
│   │   │   │   │   ├── RequirementFilters.tsx
│   │   │   │   │   └── RequirementSearch.tsx
```

```
│   │   │   │   └── RequirementDetail/
│   │   │   │       ├── RequirementInfo.tsx
│   │   │   │       ├── MatchedStakeholders.tsx
│   │   │   │       └── RequirementActions.tsx
│   │   │   ├── hooks/
│   │   │   │   ├── useRequirement.ts
│   │   │   │   └── useRequirementForm.ts
│   │   │   ├── services/
│   │   │   │   └── requirement.service.ts
│   │   │   ├── store/
│   │   │   │   └── requirementSlice.ts
│   │   │   ├── types/
│   │   │   │   └── requirement.types.ts
│   │   │   └── pages/
│   │   │       ├── CreateRequirement.tsx
│   │   │       ├── RequirementsList.tsx
│   │   │       └── RequirementDetail.tsx
│   │   │
│   │   ├── matching/
│   │   │   ├── components/
│   │   │   │   ├── AIMatchResults/
│   │   │   │   │   ├── MatchCard.tsx
│   │   │   │   │   ├── MatchScore.tsx
│   │   │   │   │   └── MatchExplanation.tsx
│   │   │   │   ├── MatchFilters/
│   │   │   │   └── MatchComparison/
│   │   │   ├── hooks/
│   │   │   │   └── useAIMatching.ts
│   │   │   ├── services/
│   │   │   │   └── matching.service.ts
│   │   │   └── types/
│   │   │       └── matching.types.ts
│   │   │
│   │   ├── rfq/
│   │   │   ├── components/
│   │   │   │   ├── CreateRFQ/
│   │   │   │   │   ├── RFQForm.tsx
│   │   │   │   │   ├── StakeholderSelection.tsx
│   │   │   │   │   └── RFQPreview.tsx
│   │   │   │   ├── QuoteManagement/
│   │   │   │   │   ├── QuoteList.tsx
│   │   │   │   │   ├── QuoteComparison.tsx
│   │   │   │   │   └── QuoteDetails.tsx
│   │   │   │   └── RFQTracking/
│   │   │   ├── services/
│   │   │   │   └── rfq.service.ts
│   │   │   └── pages/
```

```
│   │   │       ├── CreateRFQ.tsx
│   │   │       ├── RFQList.tsx
│   │   │       └── QuoteComparison.tsx
│   │   │
│   │   ├── orders/
│   │   │   ├── components/
│   │   │   │   ├── PurchaseOrder/
│   │   │   │   │   ├── POCreation.tsx
│   │   │   │   │   ├── POApproval.tsx
│   │   │   │   │   └── POTracking.tsx
│   │   │   │   ├── OrderManagement/
│   │   │   │   └── PaymentProcessing/
│   │   │   ├── services/
│   │   │   │   └── order.service.ts
│   │   │   └── pages/
│   │   │       ├── CreatePO.tsx
│   │   │       ├── OrderList.tsx
│   │   │       └── OrderDetail.tsx
│   │   │
│   │   ├── projects/
│   │   │   ├── components/
│   │   │   │   ├── ProjectTimeline/
│   │   │   │   ├── MilestoneTracking/
│   │   │   │   └── ProjectDocuments/
│   │   │   ├── services/
│   │   │   └── pages/
│   │   │
│   │   ├── profiles/
│   │   │   ├── components/
│   │   │   │   ├── industry/
│   │   │   │   ├── expert/
│   │   │   │   ├── vendor/
│   │   │   │   │   ├── service/
│   │   │   │   │   ├── product/
│   │   │   │   │   └── common/
│   │   │   │   └── logistics/
│   │   │   ├── services/
│   │   │   └── pages/
│   │   │
│   │   └── analytics/
│   │       ├── components/
│   │       │   ├── SpendAnalytics/
│   │       │   ├── VendorPerformance/
│   │       │   └── RequirementInsights/
│   │       └── pages/
│   │
│   ├── shared/
```

```
│   │   ├── components/
│   │   │   ├── layout/
│   │   │   │   ├── Header/
│   │   │   │   ├── Sidebar/
│   │   │   │   ├── Footer/
│   │   │   │   └── PageLayout/
│   │   │   ├── ui/                    # shadcn/ui components
│   │   │   │   ├── button.tsx
│   │   │   │   ├── card.tsx
│   │   │   │   ├── dialog.tsx
│   │   │   │   ├── form.tsx
│   │   │   │   ├── input.tsx
│   │   │   │   ├── select.tsx
│   │   │   │   ├── table.tsx
│   │   │   │   └── ... (other UI components)
│   │   │   ├── feedback/
│   │   │   │   ├── LoadingSpinner.tsx
│   │   │   │   ├── ErrorBoundary.tsx
│   │   │   │   └── EmptyState.tsx
│   │   │   └── common/
│   │   │       ├── DataTable/
│   │   │       ├── FileUpload/
│   │   │       └── SearchBar/
│   │   │
│   │   ├── hooks/
│   │   │   ├── useApi.ts
│   │   │   ├── useDebounce.ts
│   │   │   ├── useLocalStorage.ts
│   │   │   └── usePagination.ts
│   │   │
│   │   ├── services/
│   │   │   ├── api.service.ts
│   │   │   ├── notification.service.ts
│   │   │   └── file.service.ts
│   │   │
│   │   ├── utils/
│   │   │   ├── constants.ts
│   │   │   ├── validators.ts
│   │   │   ├── formatters.ts
│   │   │   └── helpers.ts
│   │   │
│   │   └── types/
│   │       ├── api.types.ts
│   │       ├── common.types.ts
│   │       └── global.d.ts
│   │
│   ├── config/
```

```
│   │   ├── api.config.ts
│   │   ├── app.config.ts
│   │   └── roles.config.ts
│   │
│   ├── store/
│   │   ├── index.ts
│   │   ├── hooks.ts
│   │   └── middleware/
│   │       └── api.middleware.ts
│   │
│   ├── styles/
│   │   ├── globals.css
│   │   ├── tailwind.css
│   │   └── components/
│   │
│   └── main.tsx
│
├── tests/
│   ├── unit/
│   ├── integration/
│   └── e2e/
│
├── .env.example
├── .env.development
├── .env.production
├── .gitignore
├── .eslintrc.js
├── .prettierrc
├── docker-compose.yml
├── Dockerfile
├── index.html
├── package.json
├── tailwind.config.ts
├── tsconfig.json
└── vite.config.ts
```

4. Backend Architecture (Java Spring Boot Microservices)

4.1 Complete Microservices Structure

```
diligince-backend/
├── api-gateway/
│   ├── src/main/java/com/diligince/gateway/
│   │   ├── config/
│   │   │   ├── SecurityConfig.java
│   │   │   ├── RouteConfig.java
│   │   │   └── CorsConfig.java
│   │   ├── filters/
│   │   │   ├── AuthenticationFilter.java
│   │   │   └── RateLimitFilter.java
│   │   └── GatewayApplication.java
│   └── pom.xml
│
├── auth-service/
│   ├── src/main/java/com/diligince/auth/
│   │   ├── controller/
│   │   │   ├── AuthController.java
│   │   │   └── TokenController.java
│   │   ├── service/
│   │   │   ├── AuthService.java
│   │   │   ├── TokenService.java
│   │   │   └── OAuthService.java
│   │   ├── repository/
│   │   │   └── UserCredentialRepository.java
│   │   ├── entity/
│   │   │   ├── UserCredential.java
│   │   │   └── RefreshToken.java
│   │   ├── dto/
│   │   │   ├── LoginRequest.java
│   │   │   ├── RegisterRequest.java
│   │   │   └── AuthResponse.java
│   │   ├── security/
│   │   │   ├── JwtTokenProvider.java
│   │   │   └── SecurityConfig.java
│   │   └── AuthServiceApplication.java
│   └── pom.xml
│
├── user-service/
│   ├── src/main/java/com/diligince/user/
│   │   ├── controller/
│   │   │   ├── UserController.java
│   │   │   ├── ProfileController.java
│   │   │   └── VerificationController.java
│   │   ├── service/
│   │   │   ├── UserService.java
│   │   │   ├── ProfileService.java
```

```
│   │   │   └── VerificationService.java
│   │   ├── repository/
│   │   │   ├── UserRepository.java
│   │   │   ├── IndustryProfileRepository.java
│   │   │   ├── ExpertProfileRepository.java
│   │   │   └── VendorProfileRepository.java
│   │   ├── entity/
│   │   │   ├── User.java
│   │   │   ├── IndustryProfile.java
│   │   │   ├── ExpertProfile.java
│   │   │   ├── ServiceVendorProfile.java
│   │   │   ├── ProductVendorProfile.java
│   │   │   └── LogisticsProfile.java
│   │   └── UserServiceApplication.java
│   └── pom.xml
│
├── requirement-service/
│   ├── src/main/java/com/diligince/requirement/
│   │   ├── controller/
│   │   │   └── RequirementController.java
│   │   ├── service/
│   │   │   ├── RequirementService.java
│   │   │   └── RequirementMatchingService.java
│   │   ├── repository/
│   │   │   └── RequirementRepository.java
│   │   ├── entity/
│   │   │   ├── Requirement.java
│   │   │   ├── RequirementCategory.java
│   │   │   └── RequirementDocument.java
│   │   ├── dto/
│   │   │   ├── RequirementRequest.java
│   │   │   └── RequirementResponse.java
│   │   └── RequirementServiceApplication.java
│   └── pom.xml
│
├── matching-service/
│   ├── src/main/java/com/diligince/matching/
│   │   ├── controller/
│   │   │   └── MatchingController.java
│   │   ├── service/
│   │   │   ├── AIMatchingService.java
│   │   │   ├── MatchScoringService.java
│   │   │   └── RecommendationService.java
│   │   ├── repository/
│   │   │   └── MatchResultRepository.java
│   │   ├── entity/
│   │   │   ├── MatchResult.java
```

```
│   │   │   └── MatchCriteria.java
│   │   ├── integration/
│   │   │   └── AzureAIClient.java
│   │   └── MatchingServiceApplication.java
│   └── pom.xml
│
├── rfq-service/
│   ├── src/main/java/com/diligince/rfq/
│   │   ├── controller/
│   │   │   ├── RFQController.java
│   │   │   └── QuoteController.java
│   │   ├── service/
│   │   │   ├── RFQService.java
│   │   │   ├── QuoteService.java
│   │   │   └── QuoteComparisonService.java
│   │   ├── repository/
│   │   │   ├── RFQRepository.java
│   │   │   └── QuoteRepository.java
│   │   ├── entity/
│   │   │   ├── RFQ.java
│   │   │   ├── RFQInvitation.java
│   │   │   └── Quote.java
│   │   └── RFQServiceApplication.java
│   └── pom.xml
│
├── order-service/
│   ├── src/main/java/com/diligince/order/
│   │   ├── controller/
│   │   │   ├── PurchaseOrderController.java
│   │   │   └── OrderTrackingController.java
│   │   ├── service/
│   │   │   ├── PurchaseOrderService.java
│   │   │   └── OrderTrackingService.java
│   │   ├── repository/
│   │   │   └── PurchaseOrderRepository.java
│   │   ├── entity/
│   │   │   ├── PurchaseOrder.java
│   │   │   ├── OrderItem.java
│   │   │   └── OrderStatus.java
│   │   └── OrderServiceApplication.java
│   └── pom.xml
│
├── payment-service/
│   ├── src/main/java/com/diligince/payment/
│   │   ├── controller/
│   │   │   └── PaymentController.java
│   │   ├── service/
```

```
│   │   │   ├── PaymentService.java
│   │   │   └── InvoiceService.java
│   │   ├── repository/
│   │   │   └── PaymentRepository.java
│   │   ├── entity/
│   │   │   ├── Payment.java
│   │   │   └── Invoice.java
│   │   ├── integration/
│   │   │   ├── RazorpayClient.java
│   │   │   └── PayUClient.java
│   │   └── PaymentServiceApplication.java
│   └── pom.xml
│
├── notification-service/
│   ├── src/main/java/com/diligince/notification/
│   │   ├── controller/
│   │   │   └── NotificationController.java
│   │   ├── service/
│   │   │   ├── EmailService.java
│   │   │   ├── SMSService.java
│   │   │   └── PushNotificationService.java
│   │   ├── template/
│   │   │   └── EmailTemplates.java
│   │   └── NotificationServiceApplication.java
│   └── pom.xml
│
├── project-service/
│   ├── src/main/java/com/diligince/project/
│   │   ├── controller/
│   │   │   └── ProjectController.java
│   │   ├── service/
│   │   │   ├── ProjectService.java
│   │   │   └── MilestoneService.java
│   │   ├── repository/
│   │   │   └── ProjectRepository.java
│   │   ├── entity/
│   │   │   ├── Project.java
│   │   │   └── Milestone.java
│   │   └── ProjectServiceApplication.java
│   └── pom.xml
│
├── common-lib/
│   ├── src/main/java/com/diligince/common/
│   │   ├── dto/
│   │   ├── exception/
│   │   ├── utils/
│   │   └── constants/
```

```
│   └── pom.xml
│
├── config-server/
├── service-registry/
├── docker-compose.yml
└── pom.xml (parent)
```

5. Database Schema

5.1 Core Tables

sql

```sql
-- User Management
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    phone VARCHAR(20),
    role VARCHAR(50) NOT NULL CHECK (role IN ('INDUSTRY', 'EXPERT', 'SERVICE_VENDOR', 'PRODUCT_
    is_verified BOOLEAN DEFAULT FALSE,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Industry Profiles
CREATE TABLE industry_profiles (
    user_id UUID PRIMARY KEY REFERENCES users(id),
    company_name VARCHAR(255) NOT NULL,
    industry_type VARCHAR(100),
    company_size VARCHAR(50),
    gst_number VARCHAR(50) UNIQUE,
    pan_number VARCHAR(50),
    address TEXT,
    city VARCHAR(100),
    state VARCHAR(100),
    pincode VARCHAR(10),
    website VARCHAR(255),
    description TEXT
);

-- Expert Profiles
CREATE TABLE expert_profiles (
    user_id UUID PRIMARY KEY REFERENCES users(id),
    full_name VARCHAR(255) NOT NULL,
    expertise TEXT[],
    experience_years INTEGER,
    certifications JSONB,
    education JSONB,
    hourly_rate DECIMAL(10,2),
    availability VARCHAR(50),
    linkedin_profile VARCHAR(255),
    portfolio_url VARCHAR(255),
    skills TEXT[]
);

-- Vendor Profiles (Service & Product)
CREATE TABLE vendor_profiles (
    user_id UUID PRIMARY KEY REFERENCES users(id),
```

```
    vendor_type VARCHAR(50) NOT NULL CHECK (vendor_type IN ('SERVICE', 'PRODUCT')),
    company_name VARCHAR(255) NOT NULL,
    gst_number VARCHAR(50) UNIQUE,
    pan_number VARCHAR(50),
    establishment_year INTEGER,
    employee_count VARCHAR(50),
    annual_revenue VARCHAR(50),
    service_categories TEXT[],
    product_categories TEXT[],
    certifications JSONB,
    major_clients TEXT[]
);

-- Requirements
CREATE TABLE requirements (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    industry_id UUID REFERENCES users(id),
    category VARCHAR(50) NOT NULL CHECK (category IN ('EXPERT', 'SERVICE', 'PRODUCT', 'LOGISTIC
    title VARCHAR(500) NOT NULL,
    description TEXT,
    technical_specs JSONB,
    budget_min DECIMAL(12,2),
    budget_max DECIMAL(12,2),
    currency VARCHAR(3) DEFAULT 'INR',
    timeline_start DATE,
    timeline_end DATE,
    location_city VARCHAR(100),
    location_state VARCHAR(100),
    is_urgent BOOLEAN DEFAULT FALSE,
    status VARCHAR(50) DEFAULT 'DRAFT',
    published_at TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- AI Match Results
CREATE TABLE match_results (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    requirement_id UUID REFERENCES requirements(id),
    stakeholder_id UUID REFERENCES users(id),
    match_score DECIMAL(5,2),
    skill_score DECIMAL(5,2),
    location_score DECIMAL(5,2),
    experience_score DECIMAL(5,2),
    capacity_score DECIMAL(5,2),
    price_score DECIMAL(5,2),
    ai_explanation TEXT,
```

```
    is_selected BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- RFQs
CREATE TABLE rfqs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    requirement_id UUID REFERENCES requirements(id),
    created_by UUID REFERENCES users(id),
    rfq_number VARCHAR(50) UNIQUE,
    deadline TIMESTAMP,
    terms_conditions TEXT,
    status VARCHAR(50) DEFAULT 'ACTIVE',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Quotes
CREATE TABLE quotes (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    rfq_id UUID REFERENCES rfqs(id),
    vendor_id UUID REFERENCES users(id),
    quote_number VARCHAR(50) UNIQUE,
    quoted_amount DECIMAL(12,2),
    currency VARCHAR(3) DEFAULT 'INR',
    validity_date DATE,
    delivery_timeline INTEGER, -- in days
    payment_terms TEXT,
    technical_proposal TEXT,
    commercial_proposal TEXT,
    status VARCHAR(50) DEFAULT 'SUBMITTED',
    submitted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    is_selected BOOLEAN DEFAULT FALSE
);

-- Purchase Orders
CREATE TABLE purchase_orders (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    quote_id UUID REFERENCES quotes(id),
    po_number VARCHAR(50) UNIQUE,
    industry_id UUID REFERENCES users(id),
    vendor_id UUID REFERENCES users(id),
    total_amount DECIMAL(12,2),
    tax_amount DECIMAL(12,2),
    final_amount DECIMAL(12,2),
    payment_terms TEXT,
    delivery_address TEXT,
    billing_address TEXT,
```

```
    status VARCHAR(50) DEFAULT 'DRAFT',
    approved_by UUID REFERENCES users(id),
    approved_at TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Projects
CREATE TABLE projects (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    po_id UUID REFERENCES purchase_orders(id),
    project_name VARCHAR(255),
    start_date DATE,
    end_date DATE,
    status VARCHAR(50) DEFAULT 'NOT_STARTED',
    progress_percentage INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Payments
CREATE TABLE payments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    po_id UUID REFERENCES purchase_orders(id),
    payment_number VARCHAR(50) UNIQUE,
    amount DECIMAL(12,2),
    payment_method VARCHAR(50),
    transaction_id VARCHAR(100),
    payment_date TIMESTAMP,
    status VARCHAR(50),
    gateway_response JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

6. AI Architecture

6.1 AI Matching Engine Components

python

```
# AI Service Structure
diligince-ai-engine/
├── matching_engine/
│   ├── embeddings/
│   │   ├── requirement_embedder.py
│   │   └── stakeholder_embedder.py
│   ├── scoring/
│   │   ├── skill_scorer.py
│   │   ├── location_scorer.py
│   │   ├── experience_scorer.py
│   │   └── composite_scorer.py
│   ├── recommendation/
│   │   └── recommendation_engine.py
│   └── explainer/
│       └── match_explainer.py
├── models/
│   ├── requirement_classifier.py
│   └── stakeholder_ranker.py
├── api/
│   └── matching_api.py
└── requirements.txt
```

6.2 Matching Algorithm Flow

python

```python
class Diligince Matching Engine:
    def match_requirement(self, requirement: Dict, stakeholders: List[Dict]) -> List[MatchResul
        # 1. Classify requirement
        category = self.classifier.classify(requirement)

        # 2. Generate embeddings
        req_embedding = self.requirement_embedder.embed(requirement)

        # 3. Score stakeholders
        scores = []
        for stakeholder in stakeholders:
            score = self.calculate_composite_score(
                requirement=requirement,
                stakeholder=stakeholder,
                weights={
                    'skill_match': 0.30,
                    'location_proximity': 0.20,
                    'experience_relevance': 0.20,
                    'capacity_availability': 0.15,
                    'price_competitiveness': 0.15
                }
            )
            scores.append(score)

        # 4. Rank and explain
        top_matches = self.rank_matches(scores, top_k=20)
        return self.generate_explanations(top_matches)
```

7. API Design

7.1 RESTful API Structure

yaml

```
# API Endpoints
Base URL: https://api.diligince.ai/v1

# Authentication
POST   /auth/login
POST   /auth/register
POST   /auth/refresh
POST   /auth/logout

# User Management
GET    /users/profile
PUT    /users/profile
POST   /users/verify

# Requirements
GET    /requirements                  # List with filters
POST   /requirements                  # Create new
GET    /requirements/{id}             # Get details
PUT    /requirements/{id}             # Update
DELETE /requirements/{id}             # Delete
POST   /requirements/{id}/publish     # Publish requirement

# AI Matching
POST   /requirements/{id}/match       # Get AI matches
GET    /matches/{requirementId}       # Get match results
POST   /matches/{id}/select           # Select a match

# RFQ Management
POST   /requirements/{id}/rfq         # Create RFQ
GET    /rfqs                          # List RFQs
GET    /rfqs/{id}                     # RFQ details
POST   /rfqs/{id}/invite              # Send invitations

# Quotes
POST   /rfqs/{id}/quotes              # Submit quote
GET    /rfqs/{id}/quotes              # List quotes
GET    /quotes/{id}                   # Quote details
PUT    /quotes/{id}                   # Update quote
POST   /quotes/{id}/accept            # Accept quote

# Purchase Orders
POST   /quotes/{id}/po                # Create PO from quote
GET    /purchase-orders               # List POs
GET    /purchase-orders/{id}          # PO details
PUT    /purchase-orders/{id}/approve  # Approve PO
```

```
# Projects
GET    /projects                      # List projects
GET    /projects/{id}                 # Project details
PUT    /projects/{id}/progress        # Update progress

# Payments
POST   /purchase-orders/{id}/payment  # Process payment
GET    /payments                      # Payment history
GET    /payments/{id}                 # Payment details
```

8. Technology Stack Details

8.1 Frontend Stack

Framework: React 18+ with TypeScript

Styling: TailwindCSS 3.x

State Management: Redux Toolkit

Routing: React Router v6

Forms: React Hook Form + Zod

HTTP Client: Axios with interceptors

UI Components: shadcn/ui

Charts: Recharts

Date Handling: date-fns

Testing: Jest + React Testing Library

8.2 Backend Stack

Framework: Spring Boot 3.x

Language: Java 17+

API Gateway: Spring Cloud Gateway

Service Discovery: Eureka

Config Management: Spring Cloud Config

Security: Spring Security + JWT

Database: PostgreSQL (Azure Database)

Cache: Redis (Azure Cache)

Message Queue: Azure Service Bus

File Storage: Azure Blob Storage

Search: Elasticsearch

Monitoring: Spring Actuator + Prometheus

Logging: SLF4J + Logback

8.3 AI/ML Stack

Language: Python 3.9+

Framework: FastAPI

ML Libraries: scikit-learn, TensorFlow

NLP: spaCy, Transformers

Vector DB: Pinecone/Weaviate

Azure AI: Cognitive Services

8.4 DevOps Stack

Containerization: Docker

Orchestration: Kubernetes (AKS)

CI/CD: Azure DevOps

Monitoring: Prometheus + Grafana

Logging: ELK Stack

API Docs: Swagger/OpenAPI

9. Security Architecture

9.1 Security Layers

```
1. Network Security
   - Azure WAF
   - DDoS Protection
   - Private endpoints

2. Application Security
   - JWT-based authentication
   - Role-based access control (RBAC)
   - API rate limiting
   - Input validation
   - SQL injection prevention
   - XSS protection

3. Data Security
   - Encryption at rest
   - Encryption in transit (TLS 1.3)
   - Data masking for PII
   - Audit logging

4. Compliance
   - GDPR compliance
   - SOC 2 Type II
   - ISO 27001
```

10. Deployment Architecture

10.1 Azure Infrastructure

```yaml
# Azure Resources
Resource Group: diligince-prod-rg
  ├── AKS Cluster: diligince-aks
  ├── Azure SQL: diligince-sql
  ├── Cosmos DB: diligince-cosmos
  ├── Redis Cache: diligince-redis
  ├── Storage Account: diligincestorage
  ├── Service Bus: diligince-servicebus
  ├── Application Gateway: diligince-appgw
  ├── Key Vault: diligince-keyvault
  ├── Log Analytics: diligince-logs
  └── Application Insights: diligince-insights
```

10.2 Kubernetes Deployment

yaml

```yaml
# Sample Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: user-service
  namespace: diligince
spec:
  replicas: 3
  selector:
    matchLabels:
      app: user-service
  template:
    metadata:
      labels:
        app: user-service
    spec:
      containers:
      - name: user-service
        image: diligince.azurecr.io/user-service:latest
        ports:
        - containerPort: 8080
        env:
        - name: SPRING_PROFILES_ACTIVE
          value: "prod"
        resources:
          requests:
            memory: "512Mi"
            cpu: "500m"
          limits:
            memory: "1Gi"
            cpu: "1000m"
```

11. Development Workflow

11.1 Git Branch Strategy

```
main
├── develop
│   ├── feature/auth-module
│   ├── feature/requirement-management
│   ├── feature/ai-matching
│   └── feature/payment-integration
├── release/v1.0
└── hotfix/critical-bug-fix
```

11.2 Development Phases

Phase 1: Foundation (Weeks 1-4)

Team A: Authentication & User Management

Team B: Basic UI Components & Layouts

Team C: Database Setup & Core Services

Phase 2: Core Features (Weeks 5-8)

Team A: Requirement Management

Team B: RFQ & Quote System

Team C: AI Matching Integration

Phase 3: Advanced Features (Weeks 9-12)

Team A: Purchase Orders & Projects

Team B: Payment Integration

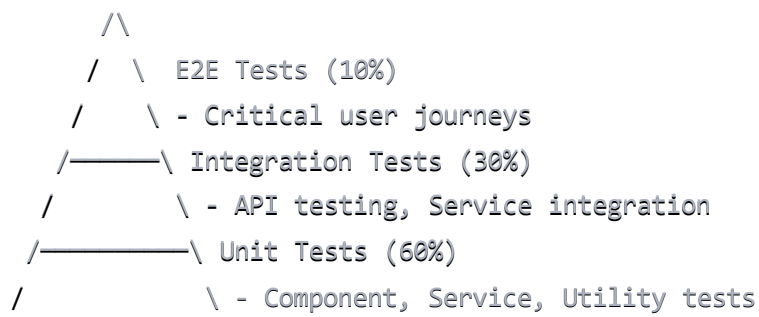
Team C: Analytics & Reporting

Phase 4: Polish & Deploy (Weeks 13-16)

All Teams: Testing, Bug Fixes, Performance Optimization

12. Testing Strategy

12.1 Testing Pyramid



12.2 Test Coverage Requirements

Frontend: Minimum 80% coverage

Backend: Minimum 85% coverage

E2E: Critical paths coverage

13. Performance Targets

Page Load: < 3 seconds

API Response: < 200ms (95th percentile)

Concurrent Users: 10,000+

Uptime: 99.9% SLA

RTO: < 1 hour

RPO: < 15 minutes

14. Success Metrics

14.1 Technical Metrics

System uptime

API response times

Error rates

User session duration

14.2 Business Metrics

User registration rate

Requirement posting frequency

RFQ-to-PO conversion rate

Platform GMV (Gross Merchandise Value)

User satisfaction score

This architecture provides a complete, scalable, and maintainable foundation for the Diligince.ai platform, supporting the vision of connecting industries with stakeholders through AI-powered matching.