Insurance Distribution Platform - Product Requirements Document & Functional Specification Document

Table of Contents

1. Product Requirements Document (PRD)
2. Functional Specification Document (FSD)
3. React Project Structure
4. API Specifications

Product Requirements Document (PRD)

1. Executive Summary

The Insurance Distribution Platform is a secure, standalone authentication and role-management system designed to support insurance sales operations through a hierarchical user management system with comprehensive security, reporting, and visualization capabilities.

2. Problem Statement

Insurance distribution companies need a secure platform that can:

- Manage complex sales hierarchies with customizable reporting lines
- Provide role-based access control with strong security measures
- Offer comprehensive reporting and analytics by role, branch, and region
- Support both web and mobile access
- Maintain complete audit trails for compliance

3. Product Vision & Goals

Vision: Create a comprehensive, secure platform that streamlines insurance distribution management through intuitive user interfaces and robust security measures.

Primary Goals:

- Provide admin-created user accounts with role-based access control
- Support customizable sales hierarchy management
- Enforce enterprise-grade security (MFA, password policies, account lockout)
- Maintain full audit trails for regulatory compliance
- Deliver responsive web and mobile-ready experience

4. Success Metrics

Metric	Target	Measurement Period
Unauthorized access incidents	< 0.5%	Annual
MFA adoption rate	> 98%	Within 1 month of launch
Admin onboarding time	≤ 2 minutes	Per new user
Report export usage rate	> 30%	Among managers monthly
System availability	99.9%	Monthly SLA
Authentication response time	< 500ms	95th percentile

5. User Personas & Roles

5.1 Primary Users

1. **Agent**
   - Primary role: Sell policies and track personal performance
   - Needs: Personal dashboard, performance metrics, supervisor visibility
   - Frequency: Daily usage
2. **Head of Branch**
   - Primary role: Manage branch-level operations and agents
   - Needs: Branch analytics, team management, reporting capabilities
   - Frequency: Daily usage
3. **Training Admin**
   - Primary role: Manage training programs and track completion
   - Needs: Training status visibility, user management within scope
   - Frequency: Weekly usage
4. **Manager, Business Development (MBD)**
   - Primary role: Manage sales teams and drive regional performance
   - Needs: Team management, performance analytics, reporting line updates
   - Frequency: Daily usage
5. **Senior Manager, Business Development (SMBD)**
   - Primary role: Oversee multiple regions and strategic operations
   - Needs: Multi-region visibility, comprehensive analytics, strategic reporting
   - Frequency: Daily usage
6. **System Admin**
   - Primary role: System configuration and user management
   - Needs: Full system access, security configuration, audit capabilities

- Frequency: As needed

6. Product Requirements

6.1 Functional Requirements

6.1.1 Authentication & Security

- **REQ-001**: Admin-only user account creation (no self-registration)
- **REQ-002**: Configurable password policy enforcement
- **REQ-003**: Multi-factor authentication (SMS, Email, TOTP)
- **REQ-004**: Account lockout after configurable failed attempts
- **REQ-005**: IP and device tracking for security monitoring
- **REQ-006**: Session management with secure token handling

6.1.2 Role Management & Hierarchy

- **REQ-007**: Role-based permissions system
- **REQ-008**: Customizable reporting line management
- **REQ-009**: Branch and region assignment
- **REQ-010**: Visibility rules based on hierarchy position
- **REQ-011**: Drag-and-drop hierarchy reassignment (admin only)

6.1.3 Dashboards & Reporting

- **REQ-012**: Role-specific dashboard layouts
- **REQ-013**: Configurable dashboard widgets
- **REQ-014**: Sales performance analytics
- **REQ-015**: Report generation (CSV, Excel, PDF)
- **REQ-016**: Scheduled report exports
- **REQ-017**: Advanced filtering capabilities

6.1.4 Organizational Visualization

- **REQ-018**: Interactive organizational chart
- **REQ-019**: Search and filter capabilities
- **REQ-020**: Zoom and expand/collapse functionality
- **REQ-021**: Export org chart as PNG/PDF

6.1.5 Audit & Compliance

- **REQ-022**: Comprehensive audit logging

- **REQ-023**: Immutable audit trail storage
- **REQ-024**: Configurable data retention policies
- **REQ-025**: GDPR-compliant data deletion workflows

6.2 Non-Functional Requirements

6.2.1 Performance

- Authentication response time: < 500ms (95th percentile)
- Org chart rendering: Support up to 5,000 nodes
- Dashboard load time: < 2 seconds
- Report generation: < 10 seconds for standard reports

6.2.2 Security

- Data encryption at rest and in transit (TLS 1.3)
- Secrets management through secure vault
- Regular security audits and penetration testing
- OWASP compliance

6.2.3 Scalability

- Horizontally scalable microservices architecture
- Stateless authentication endpoints
- Database optimization for concurrent users
- CDN integration for static assets

6.2.4 Availability

- 99.9% uptime SLA
- Disaster recovery procedures
- Automated backup systems
- Health monitoring and alerting

7. Technical Constraints

- React 18+ with TypeScript
- RESTful API architecture
- Responsive design (mobile-first approach)
- Modern browser support (Chrome 90+, Firefox 88+, Safari 14+)
- WCAG 2.1 AA accessibility compliance

8. Assumptions & Dependencies

Assumptions:

- Users have reliable internet connectivity
- Administrative staff available for user onboarding
- Existing data migration capabilities if required

Dependencies:

- Authentication service infrastructure
- Email/SMS delivery services for MFA
- Database infrastructure provisioning
- SSL certificate management

9. Risk Assessment

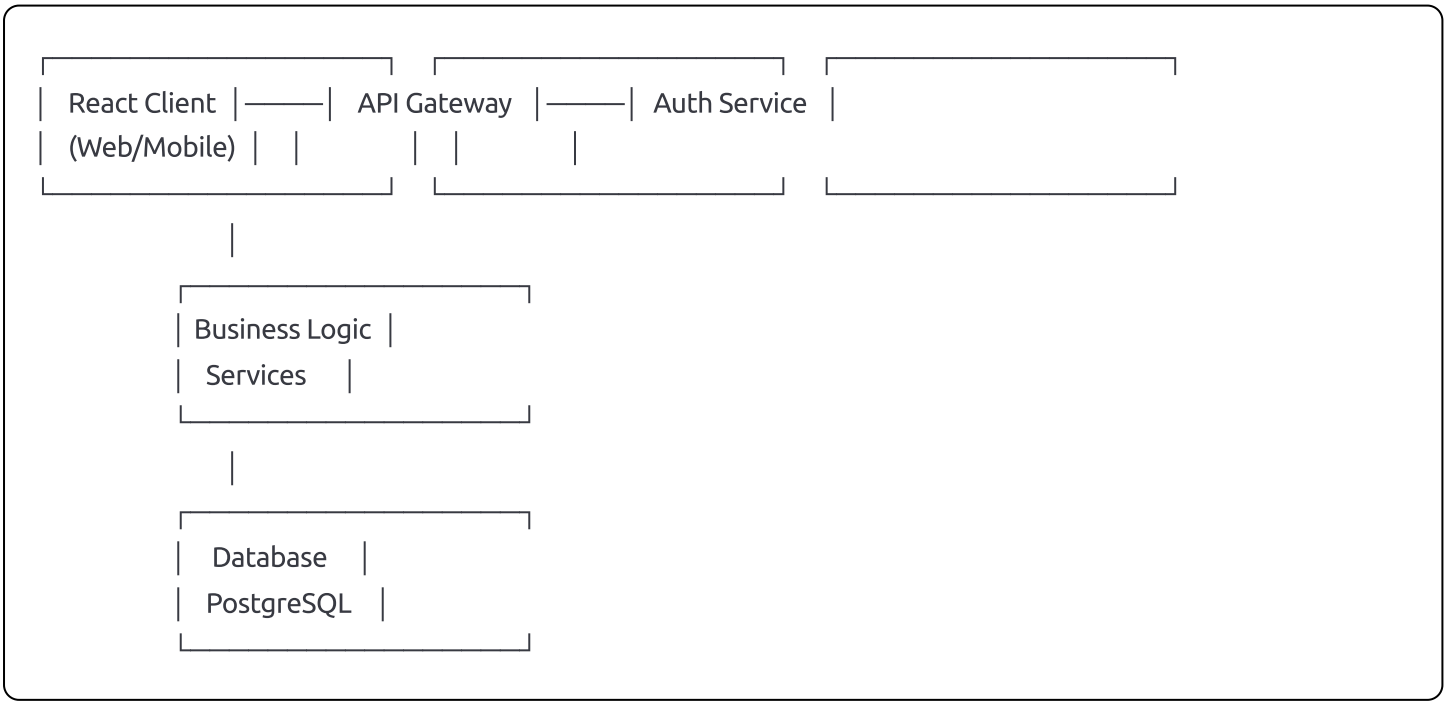
Risk	Impact	Probability	Mitigation
MFA delivery issues	High	Medium	Multiple MFA options, fallback mechanisms
Large org chart performance	Medium	Medium	Virtualization, pagination, caching
Data export misuse	High	Low	Access controls, audit logs, rate limiting
Scalability challenges	High	Medium	Microservices, horizontal scaling, monitoring

---

Functional Specification Document (FSD)

1. System Architecture

1.1 High-Level Architecture



1.2 Frontend Architecture (React)

```
src/
├── components/
│   ├── common/
│   ├── auth/
│   ├── dashboard/
│   ├── hierarchy/
│   ├── reports/
│   └── admin/
├── pages/
├── hooks/
├── services/
├── store/
├── utils/
├── types/
└── constants/
```

2. Detailed Functional Specifications

2.1 Authentication System

2.1.1 Login Flow

```typescript
interface LoginRequest {
  email: string;
  password: string;
  mfaToken?: string;
}

interface LoginResponse {
  accessToken: string;
  refreshToken: string;
  user: UserProfile;
  requiresMfa: boolean;
  mfaOptions: MfaOption[];
}
```

Process Flow:

1. User enters credentials
2. System validates credentials
3. If MFA required, prompt for MFA method selection

4. Validate MFA token
5. Generate JWT tokens
6. Redirect to role-specific dashboard

2.1.2 MFA Implementation

```typescript
interface MfaSetup {
  method: 'SMS' | 'EMAIL' | 'TOTP';
  phoneNumber?: string;
  email?: string;
  totpSecret?: string;
}
```

Supported Methods:

- SMS OTP (6-digit code, 5-minute expiry)
- Email OTP (6-digit code, 10-minute expiry)
- TOTP (30-second rotation, Google Authenticator compatible)

2.2 User Management System

2.2.1 User Data Model

```typescript
```

```typescript
interface User {
  id: string;
  email: string;
  firstName: string;
  lastName: string;
  phone?: string;
  roleId: string;
  branchId: string;
  regionId?: string;
  managerId?: string;
  status: 'ACTIVE' | 'INACTIVE' | 'LOCKED' | 'PENDING';
  mfaEnabled: boolean;
  mfaMethods: MfaMethod[];
  lastLogin?: Date;
  createdAt: Date;
  updatedAt: Date;
}

interface Role {
  id: string;
  name: string;
  permissions: Permission[];
  level: number;
}

interface Permission {
  resource: string;
  actions: string[];
  scope?: 'OWN' | 'BRANCH' | 'REGION' | 'GLOBAL';
}
```

2.2.2 Role-Based Permissions Matrix

Role	View Own Data	Manage Team	Branch Analytics	Regional Analytics	User Management	System Config
Agent	✓	-	-	-	-	-
Head of Branch	✓	Branch Only	✓	-	Branch Users	-
Training Admin	✓	Scoped	Training Only	Training Only	Training Users	-
MBD	✓	✓	✓	Limited	Team Only	-
SMBD	✓	✓	✓	✓	Regional	-
System Admin	✓	✓	✓	✓	✓	✓

2.3 Hierarchy Management

2.3.1 Hierarchy Data Model

```typescript
interface HierarchyNode {
  userId: string;
  managerId?: string;
  directReports: string[];
  level: number;
  path: string[]; // Array of user IDs from root to current
}

interface HierarchyChange {
  id: string;
  userId: string;
  oldManagerId?: string;
  newManagerId?: string;
  changedBy: string;
  timestamp: Date;
  approved: boolean;
  approvedBy?: string;
}
```

2.3.2 Org Chart Component Specifications

```typescript
```

```typescript
interface OrgChartProps {
  hierarchyData: HierarchyNode[];
  userPermissions: Permission[];
  onNodeClick: (userId: string) => void;
  onNodeDrag?: (userId: string, newManagerId: string) => void;
  searchQuery?: string;
  expandedNodes: Set<string>;
  maxDepth?: number;
}
```

Features:

- SVG-based rendering for performance
- Virtualization for large datasets (>1000 nodes)
- Zoom and pan capabilities
- Search highlighting
- Drag-and-drop reassignment (admin only)
- Export to PNG/PDF

2.4 Dashboard System

2.4.1 Dashboard Configuration

```typescript
interface DashboardWidget {
  id: string;
  type: 'CHART' | 'METRIC' | 'TABLE' | 'LIST';
  title: string;
  dataSource: string;
  filters: FilterConfig[];
  chartConfig?: ChartConfig;
  refreshInterval?: number;
  position: { x: number; y: number; w: number; h: number };
}

interface Dashboard {
  id: string;
  roleId: string;
  name: string;
  widgets: DashboardWidget[];
  layout: 'GRID' | 'MASONRY';
  isDefault: boolean;
}
```

2.4.2 Role-Specific Dashboards

Agent Dashboard:

- Personal performance metrics
- Monthly sales targets
- Pipeline status
- Training completion status

Manager Dashboard:

- Team performance overview
- Branch/region comparisons
- Top performers
- Training status summary
- Export capabilities

Admin Dashboard:

- System health metrics
- User activity summary
- Security alerts
- Audit log summaries

2.5 Reporting System

2.5.1 Report Configuration

```typescript
```

```typescript
interface ReportConfig {
  id: string;
  name: string;
  type: 'SALES' | 'PERFORMANCE' | 'TRAINING' | 'AUDIT';
  dataSource: string;
  filters: ReportFilter[];
  columns: ReportColumn[];
  groupBy?: string[];
  sortBy?: SortConfig[];
  exportFormats: ('CSV' | 'XLSX' | 'PDF')[];
}

interface ReportFilter {
  field: string;
  operator: 'EQUALS' | 'CONTAINS' | 'GREATER_THAN' | 'LESS_THAN' | 'BETWEEN';
  value: any;
  required: boolean;
}
```

2.5.2 Export Capabilities

- **CSV**: Raw data export with configurable delimiters
- **Excel**: Formatted sheets with charts and styling
- **PDF**: Professional reports with company branding
- **Scheduled Exports**: Daily/weekly automated delivery

2.6 Audit System

2.6.1 Audit Event Types

typescript

```typescript
enum AuditEventType {
  LOGIN_SUCCESS = 'LOGIN_SUCCESS',
  LOGIN_FAILURE = 'LOGIN_FAILURE',
  LOGOUT = 'LOGOUT',
  USER_CREATED = 'USER_CREATED',
  USER_UPDATED = 'USER_UPDATED',
  USER_DELETED = 'USER_DELETED',
  ROLE_CHANGED = 'ROLE_CHANGED',
  HIERARCHY_CHANGED = 'HIERARCHY_CHANGED',
  MFA_ENROLLED = 'MFA_ENROLLED',
  MFA_VERIFIED = 'MFA_VERIFIED',
  REPORT_EXPORTED = 'REPORT_EXPORTED',
  PASSWORD_CHANGED = 'PASSWORD_CHANGED',
  ACCOUNT_LOCKED = 'ACCOUNT_LOCKED',
  ACCOUNT_UNLOCKED = 'ACCOUNT_UNLOCKED'
}

interface AuditEvent {
  id: string;
  eventType: AuditEventType;
  userId?: string;
  performedBy: string;
  timestamp: Date;
  ipAddress: string;
  userAgent: string;
  metadata: Record<string, any>;
  beforeState?: any;
  afterState?: any;
}
```

3. API Endpoints Specification

3.1 Authentication Endpoints

typescript

```typescript
// POST /api/auth/login
interface LoginEndpoint {
  request: LoginRequest;
  response: LoginResponse;
  errors: {
    401: 'Invalid credentials';
    423: 'Account locked';
    428: 'MFA required';
  };
}

// POST /api/auth/mfa/verify
interface MfaVerifyEndpoint {
  request: {
    token: string;
    method: MfaMethod;
  };
  response: {
    accessToken: string;
    refreshToken: string;
  };
}

// POST /api/auth/refresh
interface RefreshEndpoint {
  request: { refreshToken: string };
  response: { accessToken: string };
}
```

3.2 User Management Endpoints

typescript

```typescript
// GET /api/users
interface GetUsersEndpoint {
  queryParams: {
    page?: number;
    limit?: number;
    role?: string;
    branch?: string;
    status?: string;
    search?: string;
  };
  response: {
    users: User[];
    pagination: PaginationInfo;
  };
}

// POST /api/users
interface CreateUserEndpoint {
  request: CreateUserRequest;
  response: User;
  permissions: ['ADMIN', 'BRANCH_MANAGER'];
}

// PUT /api/users/:id
interface UpdateUserEndpoint {
  request: UpdateUserRequest;
  response: User;
  permissions: ['ADMIN', 'SELF', 'MANAGER'];
}
```

3.3 Hierarchy Endpoints

typescript

```typescript
// GET /api/hierarchy
interface GetHierarchyEndpoint {
  queryParams: {
    rootUserId?: string;
    maxDepth?: number;
    includeInactive?: boolean;
  };
  response: HierarchyNode[];
}

// PUT /api/hierarchy/reassign
interface ReassignHierarchyEndpoint {
  request: {
    userId: string;
    newManagerId?: string;
    reason?: string;
  };
  response: HierarchyChange;
  permissions: ['ADMIN', 'AUTHORIZED_MANAGER'];
}
```

4. Security Specifications

4.1 Password Policy

typescript

```typescript
interface PasswordPolicy {
  minLength: number; // Default: 8
  maxLength: number; // Default: 128
  requireUppercase: boolean; // Default: true
  requireLowercase: boolean; // Default: true
  requireNumbers: boolean; // Default: true
  requireSymbols: boolean; // Default: true
  preventReuse: number; // Default: 5 last passwords
  expiryDays?: number; // Optional
  lockoutAttempts: number; // Default: 5
  lockoutDuration: number; // Default: 30 minutes
}
```

4.2 JWT Token Configuration

typescript

```
interface TokenConfig {
  accessTokenExpiry: number; // 15 minutes
  refreshTokenExpiry: number; // 7 days
  algorithm: 'RS256';
  issuer: string;
  audience: string;
}
```

5. Performance Specifications

5.1 Frontend Performance Targets

- Initial page load: < 2 seconds
- Dashboard rendering: < 1 second
- Org chart (up to 1000 nodes): < 3 seconds
- Report generation UI: < 500ms
- Search results: < 300ms

5.2 Backend Performance Targets

- Authentication: < 200ms
- User queries: < 500ms
- Hierarchy queries: < 1 second
- Report generation: < 10 seconds
- Audit log queries: < 2 seconds

6. Testing Strategy

6.1 Unit Testing

- Component testing with React Testing Library
- Service layer testing with Jest
- Utility function testing
- Custom hooks testing

6.2 Integration Testing

- API integration testing
- Authentication flow testing
- Role-based access testing
- Report generation testing

6.3 E2E Testing

- User journey testing with Playwright
- Cross-browser compatibility
- Mobile responsive testing
- Performance testing

7. Deployment & Infrastructure

7.1 Environment Configuration

- **Development**: Local development with Docker
- **Staging**: Pre-production environment for testing
- **Production**: Multi-region deployment with load balancing

7.2 CI/CD Pipeline

1. Code commit triggers automated tests
2. Security scanning (SAST/DAST)
3. Build optimization and bundling
4. Deployment to staging environment
5. Automated E2E testing
6. Production deployment approval
7. Health checks and monitoring

8. Monitoring & Alerting

8.1 Application Monitoring

- Real-time performance metrics
- Error tracking and reporting
- User activity monitoring
- Security event monitoring

8.2 Infrastructure Monitoring

- Server health and performance
- Database performance
- Network latency
- Storage utilization

This comprehensive PRD and FSD provides the foundation for building a robust, secure, and scalable insurance distribution platform using React and modern web technologies.