RAPIDCLAIMPRO PROJECT REPORT

1. PROJECT INTRODUCTION

RapidClaimPro is an incident management and claims processing platform for reporting, tracking, and resolving incidents. It supports both **user-reported incidents** and **automatic ingestion from open data points**.

Core purpose: streamline incident workflows with role-based access, real-time updates, media management, billing, and integrations with public open data sources.

Key characteristics:

- Multi-tenant (organization isolation)
- Real-time updates via WebSocket
- Role-based access control (Victim, Inspector, Admin)
- Payment processing (Stripe)
- GDPR-compliant data handling
- Scalable architecture (Firebase, Node.js, React)

---

2. USE CASES

2.1 Victim/Claimant Use Cases

- Report incidents with details, location, and media
- Track status in real time
- Receive notifications on updates and assignments
- View incident history
- Upload evidence (photos, videos, documents)

2.2 Inspector/Claims Professional Use Cases

- View and filter incident dashboard
- Receive assignments and notifications
- Update status and progress
- Assign cases to team members
- Access incident details and media
- Generate reports and analytics
- Manage workload

2.3 Administrator Use Cases

- Manage users and roles
- Configure open data point integrations
- Monitor data feeds and sync status
- View analytics (revenue, subscriptions, credit usage)
- Manage billing (plans, credits, add-ons)
- System health monitoring
- GDPR requests (export, deletion)

2.4 Open Data Aggregation Use Cases

- Automatically ingest incidents from open data APIs
- Normalize data from multiple sources to a unified schema
- Schedule periodic syncs
- Broadcast new incidents in real time
- Monitor data quality
- Track aggregation statistics

2.5 Billing & Payment Use Cases

- Subscription plans (Starter, Professional, Enterprise)
- Pay-per-incident via credit bundles
- Add-on purchases (AI Summaries, API Access, White-Label)
- Payment history and invoices
- Credit balance tracking
- Usage-based billing

2.6 Real-Time Communication Use Cases

- Live incident updates
- Assignment notifications
- Open data feed streaming
- System-wide announcements
- Connection status monitoring

---

3. ARCHITECTURE

3.1 High-Level Architecture

```
RapidClaimPro/
├── shared/          # Shared TypeScript types and interfaces
├── web/             # React frontend (Vite + TypeScript + 
TailwindCSS)
├── server/          # Node.js backend (Express + TypeScript + 
Firebase Admin)
└── firestore.rules  # Firebase security rules
```

3.2 Technology Stack

Frontend:

- React 18 with TypeScript

- Vite
- TailwindCSS
- React Query
- React Router
- Firebase Auth
- Socket.IO Client

Backend:

- Node.js with TypeScript
- Express.js
- Firebase Admin SDK
- Firestore (database)
- Firebase Storage (media)
- Socket.IO (WebSocket)
- Stripe (payments)

Shared:

- TypeScript types and interfaces
- Shared business logic

3.3 Backend Architecture

```
server/
├── src/
│   ├── index.ts          # Server entry point
│   ├── app.ts            # Express application setup
│   ├── firebase.ts          # Firebase Admin SDK initialization
│   ├── types/               # TypeScript type definitions
│   ├── middleware/          # Custom middleware
│   │   ├── auth.ts          # JWT authentication
│   │   ├── validation.ts     # Request validation
│   │   ├── errorHandler.ts   # Error handling
│   │   └── stripeWebhook.ts  # Stripe webhook verification
│   ├── services/            # Business logic services
│   │   ├── incidents.ts      # Incident management
│   │   ├── users.ts         # User management
│   │   ├── media.ts         # Media handling
│   │   ├── notifications.ts  # Notification system
│   │   ├── billing.ts       # Billing and credits
│   │   ├── aggregator.ts     # 911 data aggregation
│   │   └── realtime.ts      # WebSocket service
│   ├── routes/              # API route handlers
│   │   ├── incidents.ts      # Incident endpoints
│   │   ├── users.ts         # User endpoints
│   │   ├── media.ts         # Media endpoints
│   │   ├── notifications.ts  # Notification endpoints
│   │   ├── admin.ts         # Admin endpoints
│   │   ├── billing.ts       # Billing endpoints
│   │   ├── aggregator.ts     # Aggregator endpoints
│   │   └── realtime.ts      # Real-time endpoints
│   ├── controllers/         # Request handlers
│   ├── aggregator/          # 911 Aggregator Engine
│   │   ├── core/
│   │   │   └── AggregatorEngine.ts  # Core aggregation engine
│   │   └── plugins/
│   │       ├── ArcGISPlugin.ts      # ArcGIS integration
│   │       ├── CKANPlugin.ts        # CKAN integration
│   │       └── SocrataPlugin.ts     # Socrata integration
│   └── validation/          # Zod schemas
```

3.4 Frontend Architecture

```
web/
├── src/
│   ├── App.tsx              # Main application component
│   ├── main.tsx             # Application entry point
│   ├── components/          # Reusable components
│   │   ├── incident/        # Incident-related components
│   │   ├── layout/          # Layout components
│   │   └── ui/              # UI components
│   ├── pages/               # Page components
│   │   ├── Dashboard.tsx
│   │   ├── IncidentFeed.tsx
│   │   ├── AdminDashboard.tsx
│   │   ├── BillingDashboard.tsx
│   │   └── Pricing.tsx
│   ├── hooks/               # Custom React hooks
│   │   ├── useRealtimeIncidents.tsx
│   │   ├── useWebSocket.ts
│   │   └── useStripe.ts
│   ├── lib/              # Utility libraries
│   │   ├── api.ts         # API client
│   │   ├── auth.tsx        # Authentication
│   │   ├── billingService.ts # Billing API client
│   │   └── firebase.ts      # Firebase configuration
│   └── styles/            # CSS styles
```

3.5 Data Architecture

Firestore collections:

- users
- organizations
- incidents
- assignments
- media
- notifications
- audit_logs
- data_sources
- subscriptions
- creditBalances
- creditTransactions
- addOns

- payments
- invoices

3.6 Security Architecture

- Firebase ID Token verification
- JWT-based authentication
- Role-based access control (RBAC)
- Organization isolation
- Firebase Security Rules
- Helmet.js, CORS, rate limiting, and Zod validation
- Stripe webhook signature verification

3.7 Open Data Aggregator Engine Architecture

Plugin-based system:

- Core engine for data ingestion from open sources
- DataSourcePlugin interface for different data formats
- Unified schema normalization
- Scheduled aggregation and real-time broadcasting
- Data quality validation

Workflow:

1. Configure open data source (admin)
2. Initialize plugin
3. Fetch data from source API

4. Normalize to unified schema
5. Store in Firestore
6. Broadcast updates to clients

3.8 Real-Time Communication Architecture

- WebSocket (Socket.IO) server
- JWT authentication
- Room-based broadcasting
- Event types: incident updates, assignments, data feed updates, notifications
- Connection management and health checks

3.9 Billing Architecture

- Stripe for subscriptions, credits, and add-ons
- Firestore for billing and usage records
- Webhook event handling for payments
- Real-time credit balance updates

---

4. FUTURE PLANS: EXPANSION OF OPEN DATA INTEGRATION

4.1 Expanded Data Source Integration

Current support:

- Open data portals

Planned additions:

- Broader open data points across regions and sectors
- Data partnerships with public safety and municipal platforms
- RSS/Atom feeds for emergency services
- Webhook integrations for direct data pushes
- Support for custom APIs from city or organization endpoints

---

4.2 Enhanced Data Processing

Planned features:

- Machine learning for classification and prioritization
- Natural language processing for extracting key details
- Geospatial clustering and analysis
- Data enrichment (weather, traffic, demographics)
- Data quality validation and anomaly checks

4.3 Real-Time Streaming Enhancements

- Improved WebSocket streaming and filtering
- Historical replay options
- Server-Sent Events (SSE) support
- Rate limiting and monitoring

4.4 Data Quality & Monitoring

- Quality dashboard with metrics and alerts

- Source health and uptime tracking
- Configurable validation rules
- Data lineage tracking and reporting

4.5 Integration Architecture Enhancements

- Plugin marketplace for third-party integrations
- Configuration UI for managing data connections
- Unified API gateway
- ETL pipeline for transformation and enrichment
- Caching and queuing for high performance
- Data warehouse integration for analytics
- Backup and disaster recovery automation

4.6 Advanced Features

- Multi-region deployment
- Organization-level data federation
- Custom schemas per organization
- Export tools (CSV, JSON, XML)
- REST API and webhook access
- External system synchronization

4.7 Compliance & Security

- Enhanced data encryption (at rest and in transit)
- SOC 2 and HIPAA compliance roadmap
- Full audit logging
- Automated data retention and anonymization policies

4.8 Performance & Scalability

- Horizontal scaling and load balancing
- Optimized database queries and caching
- CDN for faster media delivery
- Advanced performance monitoring

4.9 Timeline & Implementation Strategy

- Add 5–10 new open data source plugins
- Enhanced data quality monitoring
- Improved error handling and retry logic
- Basic ML classification models

CONCLUSION

RapidClaimPro delivers a scalable, real-time incident management and claims platform integrating user-generated and open data source inputs. Its modular architecture, billing capabilities, and real-time communication make it a powerful solution for modern incident management.

Future focus: diversify open data integrations, enhance automation and analytics, and scale the infrastructure for enterprise-grade reliability.