APEX

SOFTWARE REQUIREMENTS SPECIFICATION

BMW IT HUB

COS 301 | University of Pretoria

FIRE-FIGHTER ACCESS MANAGEMENT PLATFORM

APEX

Contents

Introduction

The **FireFighter Access Management Platform** provides a structured, auditable solution for emergency privilege escalations, enabling rapid incident response with strict security controls.

Business Need

In BMW's complex IT environment, delays in granting elevated access during emergencies can lead to:

- Extended downtime in manufacturing operations
- Delayed resolution of critical security vulnerabilities
- Disruption to supply chain and logistics systems
- Operational inefficiencies and increased costs

Demo 1 Scope

In Demo 1 (28 May 2025), we present the front-end design with mocked data only. The demo covers polished UI flows for:

- **Login & Registration** (via Firebase Authentication)
- **Client-side Form Validation**
- **Theme Switching** (light/dark modes)
- **Dashboard Layout**
- **Notifications Page**
- **Ticket Requests Page**

User Stories / User Characteristics

User Characteristics

1. IT Engineer

- **Goal**: Resolve urgent system issues by obtaining temporary elevated access.
- **Tech Proficiency**: High; familiar with internal ticketing systems and command-line interfaces.
- **Pain Points**: Delays in access approval during emergencies; unclear access tracking.

2. System Administrator

- **Goal**: Approve or reject FireFighter access requests and monitor sessions.
- **Tech Proficiency**: High; handles role assignments, logs, and compliance.
- **Pain Points**: Need for fast but secure access control, clear auditability, and minimal manual effort.

3. Security Auditor

- **Goal**: Investigate incidents, verify system integrity, and analyze user activity.
- **Tech Proficiency**: Moderate to high; focuses on system logs and audit trails.
- **Pain Points**: Lack of centralized logging, ambiguity in change attribution.

4. Manager / Incident Supervisor

- **Goal**: Be informed when emergency access is activated and ensure it's justified.
- **Tech Proficiency**: Medium; reads notifications, may not use CLI.
- **Pain Points**: Lack of visibility, excessive manual reporting.

User Stories

FireFighter Role Request Flow

1. **As an IT Engineer**, I want to request elevated access using a valid ticket ID so that I can fix critical issues during emergencies.
2. **As an IT Engineer**, I want the elevated role to automatically expire after a set time so that I don't risk forgetting to revoke it.
3. **As a System Administrator**, I want to receive real-time notifications when a FireFighter request is submitted so that I can respond quickly.

Approval and Governance

4. **As a System Administrator**, I want to approve or reject FireFighter requests based on predefined conditions so that access is controlled and auditable.
5. **As a Security Auditor**, I want to view a complete list of past FireFighter sessions and the actions performed so that I can ensure compliance.
6. **As a System Administrator**, I want to revoke access manually in critical cases so that I can maintain system integrity.

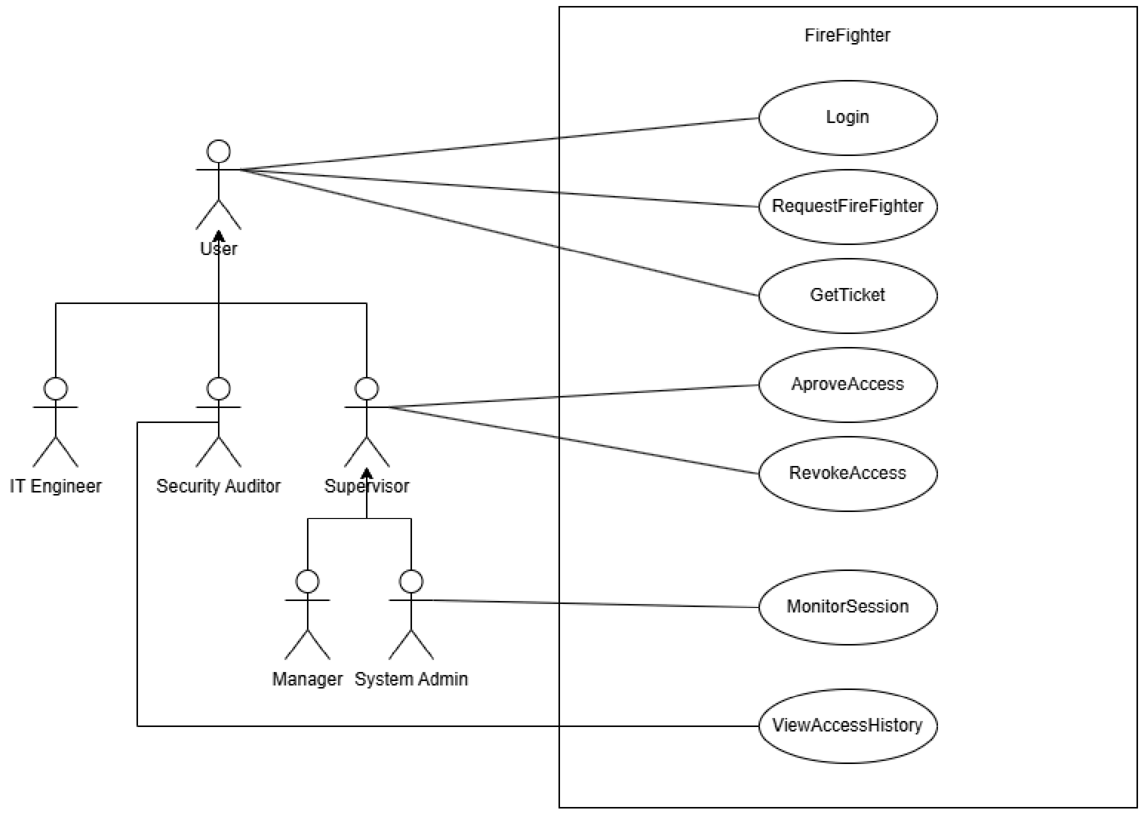
Notification & Monitoring

7. **As a Manager**, I want to receive a summary of FireFighter access events in my email or chat so that I stay informed about ongoing emergencies.
8. **As a User**, I want to get notified when my access is about to expire so that I can extend or wrap up my actions.

Auditing & Logging

9. **As a Security Auditor**, I want every change made during FireFighter access to be logged with user and timestamp so that actions can be traced.
10. **As a System Administrator**, I want a dashboard of ongoing FireFighter sessions and their statuses so that I can monitor live activity.

Use Case Diagram



Functional Requirements

1.1. User Authentication and Registration

1.1.1. The system will allow users to register for an account using a secure, email-based sign-up process.

1.1.2. The system will allow registered users to log in securely using their credentials.

1.1.3. The system will maintain user session state across both web and mobile interfaces.

1.1.4. The system will enforce email verification or SSO before granting access to sensitive features.

1.2. Role Request Management

1.2.1. Allow authorized users to make/send a request for temporary FireFighter access by entering a valid ticket ID.

1.2.2. The system shall verify that the requesting user is authorized to request extended access.

1.2.3. The ticket ID used in a request will be validated before access is granted.

1.2.4. The system will notify the user whether the access they requested was approved or denied.

1.3. Time-Bound access control

1.3.1. The system will grant FireFighter access for a set duration as specified in the application settings.

1.3.2. The system shall remove the FireFighter access once the access duration expires.

1.3.3. The system will allow administrators to manually revoke active FireFighter roles before the expiry time.

1.3.4. The systems will allow users to terminate their own access early.

1.4. Action Attribution and Logging

1.4.1. The system will record all actions performed under an active FireFighter role session.

1.4.2. For each action performed, the system will store the user responsible, the ticketID associated with the session, and the timestamp of the action.

1.4.3. The system will ensure that logs are stored securely so that the logs cannot be tampered with.

1.5. Notification system

1.5.1. The system shall send a notification to administrators when a FireFighter role request is submitted.

1.5.2. The system will notify administrators when a FireFighter role is granted and becomes active.

1.5.3. The system will notify administrators when a FireFighter role is revoked.

1.5.4. The system will provide confirmation notifications to users during request, activation and revocation.

1.6. Security and Access control

1.6.1. The system will enforce role-based access control to restrict access to sensitive operations.

1.6.2. The system shall authenticate all users using FireBase.

1.6.3. The system will log all failed access attempts, providing relevant context for security auditing.

1.6.4. The system will assign administrator roles through secure configuration.

1.6.5. Only users with the administrator role will access administrative features such as role revocation, session monitoring, and audit logs.

1.7. Admin Dashboard

1.7.1. The system will provide an interface for administrators to view all pending, active, and expired FireFighter sessions.

1.7.2. The system will allow filtering and searching through historical logs for auditing.

1.8. Ticketing System Integration

1.8.1. The system will support integration with an external ticketing system to retrieve and validate ticket details.

1.8.2. The system will fall back to a mock ticketing system if no external system is integrated.

1.9. Chatbot Interface

1.9.1. The system will provide a chatbot interface which will allow users to request and manage FireFighter roles via natural language.

1.9.2. The chatbot will support access through platforms like a web-based chat, Microsoft Teams, or Slack.

1.9.3. The chatbot will guide users through the request process using contextual prompts.

1.9.4. The chatbot will allow users to query the current status of their FireFighter role.

Service Contracts

This section defines the service contracts for the FireFighter Access Management Platform, focusing on interaction protocols between the **Angular/Ionic front-end**, **Spring Boot back-end**, **Firebase Authentication**, **PostgreSQL DB**, and future **Jira** or mock ticketing integrations. It ensures consistency, security, and traceability across all components.

1. Authentication Service (Firebase OAuth)

Provider: Firebase Authentication
Used By: Angular/Ionic Frontend
Purpose: Secure sign-in via SSO/OAuth2 and ID token retrieval.

Endpoint Behaviour

Feature	Details
Auth Provider	Firebase OAuth2
Token Format	JWT (Firebase ID Token)
Header	Authorization: Bearer <ID_TOKEN>
Expiry	Set by Firebase (usually 1 hour)
Verification	Handled server-side via Firebase Admin SDK

Example Usage (Client-side)

```
const token = await user.getIdToken();

fetch('/api/access-request', {

  method: 'POST',

  headers: {

    Authorization: `Bearer ${token}`

  },

  body: JSON.stringify({...})

});
```

2. Access Request Service

Endpoint: POST /api/access-request
Description: Allows authorised users to request FireFighter access.
Authorization: Required (Firebase ID Token)

Request JSON

```
{
  "ticketId": "INC123456",
  "durationMinutes": 30,
  "justification": "Urgent patch on DB cluster"
}
```

Response JSON

```
{
  "requestId": "REQ-001",
  "status": "PENDING",
  "expiry": "2025-05-26T12:00:00Z"
}
```

Error Handling

Code	Reason
400	Missing or invalid fields
401	No/invalid token
403	User not authorised to request
409	Duplicate active request
500	Server error

3. Role Activation & Expiry

Endpoint: PUT /api/access-request/{id}/approve
Used by: Admin roles
Effect: Activates FireFighter role for approved user

Automatic Revocation

- Role revoked automatically after duration
- Manual PUT /api/access-request/{id}/revoke for early deactivation

4. Audit Logging Service

Tech: ELK Stack (Elasticsearch, Logstash, Kibana)
Data Stored:

- Who requested access
- Ticket ID
- Access time window
- Actions performed
- IP address / device info

Format (Sample Log Document)

```
{
  "userId": "user@example.com",
  "ticketId": "INC123456",
  "action": "APPROVE_ACCESS",
  "timestamp": "2025-05-26T10:12:00Z",
  "requestId": "REQ-001",
  "role": "FIREFIGHTER"
}
```

5. Notification Service

Triggers:

- New access request submitted
- Request approved/denied
- Role expired or manually revoked

Delivery: Firebase Cloud Messaging or email (via SMTP proxy if implemented)
Example Payload:

```
{
  "type": "ROLE_GRANTED",
  "recipient": "admin@bmw.com",
  "message": "FireFighter role granted to user@example.com for 30 mins"
}
```

6. Ticket Validation

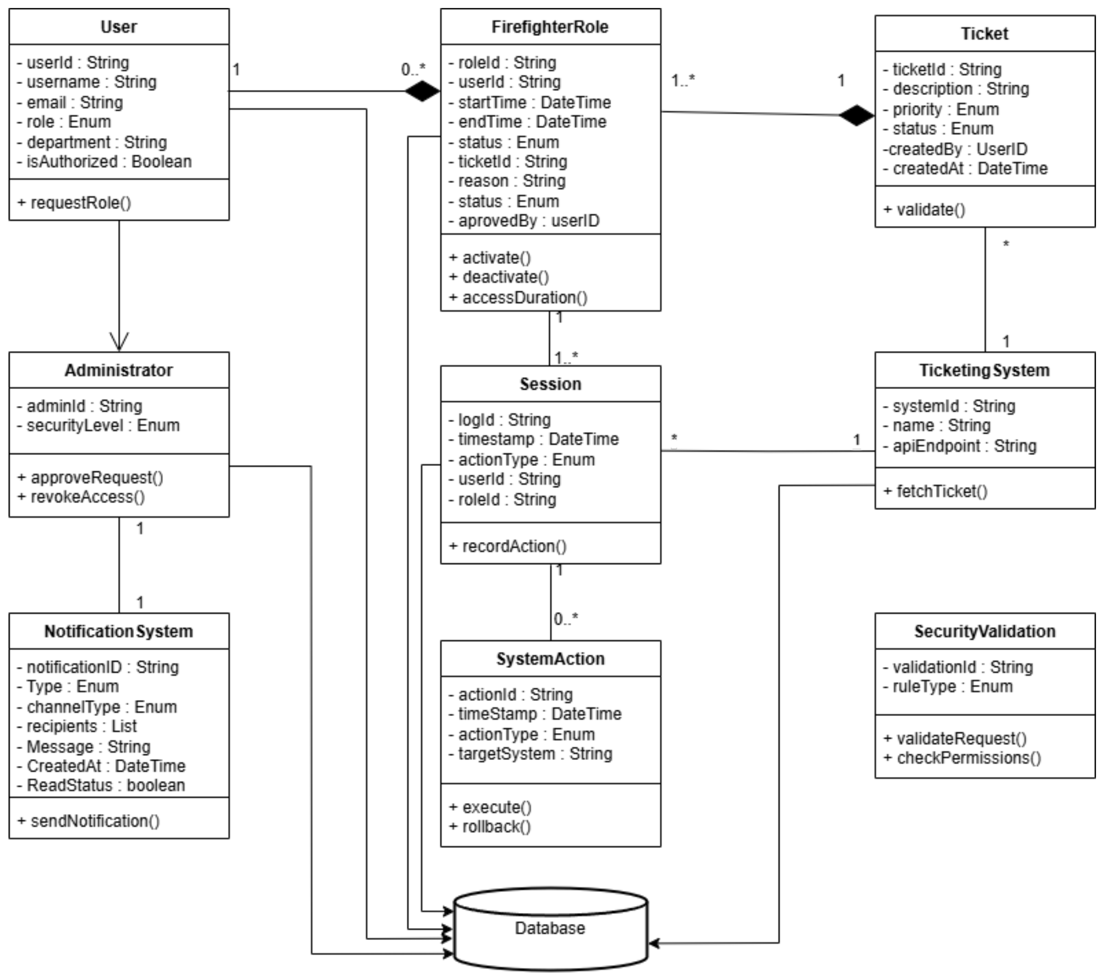
Source: Mock System or Jira Integration
Endpoint: GET /api/ticket/{ticketId}
Response:

```
{
  "ticketId": "INC123456",
  "summary": "DB node failure",
  "status": "Open",
  "valid": true
}
```

7. Technology Contract Summary

Component	Interface Type	Protocol	Notes
Frontend (Ionic)	RESTful API	HTTPS	Calls Spring Boot API via NGINX proxy
Backend (Spring)	REST Controller	HTTPS + JWT	Verifies Firebase tokens
DB (PostgreSQL)	JDBC / ORM	Localhost	Hibernate ORM
Logs	Logstash JSON	Filebeat	Streamed to Elasticsearch
CI/CD	GitHub Actions	YAML	Auto-build/test/deploy containers

Domain Model



This figure illustrates key domain entities and their relationships in the FireFighter platform:

- **User & Administrator**
  A User may request time-limited roles; an Administrator approves and manages those roles.
- **Ticket & TicketingSystem**
  Each emergency access request is linked to a Ticket, which is created and tracked by the TicketingSystem.
- **FirefighterRole & Session**
  When a Ticket is approved, a FirefighterRole is bound to the User and materialised as an active Session with a defined expiry.
- **SystemAction & SecurityValidation**
  Every privileged operation performed during a Session is modelled as a SystemAction and subject to SecurityValidation to ensure policy compliance.

- **NotificationSystem**
  All lifecycle events (request, activation, revocation) are published to the NotificationSystem for real-time alerts.
- **Database**
  All domain objects—Tickets, Sessions, Actions, Users, Roles—are persisted in the central Database.

Architectural Requirements

Quality Requirements (Non-Functional Requirements)

Quality Attribute	Description
Security	Highest priority: enforce role-based access control (RBAC), encryption (TLS), and audit logging.
Availability	System must be highly available for emergency access; implement failover and retries.
Scalability	Handle multiple concurrent requests across departments or environments without degradation.
Audibility	Every action must be tracked, timestamped, and attributable to a user.
Performance	Access requests and approvals must complete in under 1–2 seconds under normal load.
Reliability	Automatic revocation and session management must never fail silently.
Maintainability	Codebase must support CI/CD, unit testing, and clear separation of concerns.
Usability	Interface must guide users intuitively during critical situations.
Interoperability	Must support integration with ticketing systems, Active Directory, and monitoring tools.
Modifiability	Future roles or approval chains must be easy to add without full system redesign.

Architectural Patterns

Pattern	Justification
Layered Architecture	Clear separation of concerns: Presentation, Business Logic, Persistence, Security.
Microkernel (Plugin)	Useful if you later need to add custom access validation rules or plugins for new tools.
Client-Server	Frontend (Angular + Ionic) interacts with backend (Spring Boot REST API).
Event-Driven Architecture	Useful for logging, notifications, and revocations (e.g., WebSocket/Queue on access events).
Cloud-Native	Designed for containerized deployment on Docker/Kubernetes using CI/CD pipelines.

Design Patterns

Security & Access Control

- **Role-Based Access Control (RBAC)**
  Assign permissions based on user roles: Engineer, Admin, Auditor, etc.
- **Proxy Pattern**
  Control access to sensitive operations during FireFighter sessions.
- **Façade Pattern**

Monitoring & Logging

- **Observer Pattern**
  Notify relevant users/admins when requests or sessions are updated (e.g., status change).
- **Chain of Responsibility**
  For approval logic: e.g., if Admin 1 is unavailable, escalate to Admin 2.

Business Logic

- **Strategy Pattern**
  Apply different session expiration strategies or validation rules (e.g., based on ticket type).
- **Builder Pattern**
  For constructing access request objects in a consistent, validated format.
- **Singleton Pattern**
  For centralized access to the audit logger or notification dispatcher.

Constraints

Constraint	Impact
Security Compliance	Must meet BMW's security policies—requires encrypted communication, OAuth2/SSO, and full audit trail.
Technology Stack Fixed	Spring Boot + Angular + PostgreSQL + Docker/Kubernetes are required per tender proposal.
Cloud Free Tier	Must run within AWS free-tier limits (limits CPU/memory scaling and service use).
CI/CD Learning Curve	Team has limited experience with GitHub Actions and cloud deployment—adds ramp-up time.
Time-Bound Access	Must implement time-based access expiration as core functionality.
Integration with Jira	Optional, but encouraged—requires mock API or actual Jira integration via REST.

Technical Requirements

This section specifies the environments, version constraints, performance/security targets, data-management policies, reliability, scalability, monitoring, and CI/CD workflow for Demo 1 and beyond.

(All of the following requirements are subject to change and are not final.)

Supported Environments

- **Desktop Browsers:** Chrome, Firefox, Edge (latest two major versions)
- **Mobile Browsers:** Safari on iOS 14+, Chrome on Android 11+
- **Operating Systems:** Windows 10+, macOS 10.14+, Ubuntu 20.04 LTS
- **Development Hosts:** Windows, macOS, Linux (for local builds and CI runners)

Platform & Version Constraints

- **Node.js & npm:** ≥ 16.x / npm ≥ 8.x
- **Angular & Ionic Capacitor:** Angular LTS; Ionic Capacitor 4.x
- **Java & Spring Boot:** OpenJDK 11+; Spring Boot 2.6+
- **Database:** PostgreSQL v13; Hibernate ORM 5.6+
- **Containerisation:** Docker 20.10+ (for local dev and future services)
- **CI Runner:** GitHub Actions (ubuntu-latest)

Deployment & Infrastructure

- **Hosting:** AWS Free-Tier EC2 (e.g., t2.micro) instances for all application tiers
- **Backup Strategy:** Weekly AMI snapshots of EC2 instances

Security & Compliance

- **Authentication & User Management:** Firebase Authentication (OAuth/SSO via Google, Microsoft, etc.) with built-in user directory and token handling
- **Authorization:** Role-based rules enforced client-side and via Spring Security guards
- **Transport Security:** TLS 1.2+ for all endpoints
- **OWASP Top 10:** CSP headers, XSS/CSRF protections in Angular, secure cookie flags

Performance Requirements

- **First-Paint:** ≤ 2 s on a 4G connection
- **UI Responsiveness:** Interactive actions (theme switch, form validation) ≤ 200 ms
- **Mock Data Fetch:** ≤ 100 ms round-trip for local API stubs
- **Capacity Target:** Architected for ≥ 1 000 concurrent users in future back-end demos

Data Management

- **Demo 1:** In-memory JSON fixtures only
- **Future Production:**
  - **Backups:** Nightly PostgreSQL dump to S3
  - **Retention:**
    - **Database:** 30 days
    - **Audit Logs:** 90 days
  - **Encryption:** AES-256 at rest (via RDS or disk-level encryption)

Reliability & Availability

- **Uptime Target:** 99.9% over any 30-day period (post-Demo 1)
- **Health Checks:** Automated smoke tests of critical UI routes and mock services
- **Error Handling:** Graceful fallback screens with “retry” options

Scalability & Capacity Planning

- **Stateless Front-end:** Served from EC2; horizontal scaling by adding EC2 instances behind an ELB
- **Session Management:** Firebase tokens stored client-side; no server affinity required

Monitoring & Logging

- **Centralised Logging:** ELK Stack on EC2 (Elasticsearch, Logstash, Kibana)
- **Metrics & Alerts:**
    - **Error Rate:** Alert if > 1% of UI/API calls fail over 5 min
    - **Latency:** Alert if average API response > 500 ms over 10 min
- **Audit Trail:** Immutable Firebase “last sign-in” logs plus application activity logs

CI/CD & Quality Gates

- **Pipeline Stages:**
    1. Build & unit tests (Angular CLI; JUnit/Mockito)
    2. Static analysis (ESLint, Checkstyle), security scans (Snyk)
    3. End-to-end tests (Cypress on mock data)
    4. Docker image build & push to AWS ECR
- **Quality Gates:**
    - **Coverage:** ≥ 80%
    - **Linting:** Zero errors
    - **Vulnerabilities:** No high/critical findings

Compliance & Standards

- **Data Protection:** GDPR compliance for user data
- **Security Framework:** ISO/IEC 27001 alignment for access management
- **Documentation:**
    - **API:** Swagger UI (OpenAPI v3)
    - **Runbooks:** Versioned Markdown for deployment, rollback, and recovery