Product Requirements & Specification Document

Project Name

PropSecure - Real Estate Property Security Platform

Description

PropSecure is a secure, open-source backend platform for managing real estate property listings, user roles, and access logs. It leverages Spring Security with JWT, AOP-based logging, MySQL persistence, asynchronous background checks, and Dockerized deployment. An Angular frontend serves agents and admins.

---

1. Goals & Objectives

Goal	Description
Secure Property Management	Enable secure CRUD operations for property listings
Role-Based Access	Restrict features based on user roles (Admin, Agent)
Comprehensive Logging	Log all critical actions and access events
Async Background Checks	Perform property/agent background checks asynchronously
Modern, Futuristic UI	Provide a responsive Angular frontend for agents and admins
Open-Source & Containerized	Ensure code is open-source and easily deployable via Docker

---

2. Core Features

Feature	Description
User Authentication	JWT-based login/logout, registration, password reset
Role Management	Admins manage user roles (Admin, Agent)
Property Listings	CRUD operations on property data
Access Logging	AOP logs for all sensitive actions (CRUD, login, role changes)
Async Background Checks	Trigger and monitor background checks for properties/agents
Audit Trail	Viewable logs of all access and changes
Angular Frontend	Responsive UI for property management, user roles, and logs
Dockerized Deployment	All services and frontend deployable via Docker

---

3. User Roles & Permissions

Role	Permissions
Admin	Full access: manage users, roles, properties, view all logs, trigger checks
Agent	Manage own properties, view own logs, request background checks

---

4. Technical Specifications

4.1 Backend

Aspect	Specification
Language	Java 17+
Framework	Spring Boot (Spring Security, Spring Data JPA, AOP)
Auth	JWT (stateless, token-based)
Database	MySQL 8+
Async Processing	Spring @Async, ExecutorService
Logging	AOP for method-level logging, persisted to DB
Build Tool	Maven
Containerization	Dockerfile, docker-compose for multi-service orchestration

4.2 Frontend

Aspect	Specification
Framework	Angular 16+
Auth	JWT token storage (HTTP-only cookies/localStorage)
UI	Material Design, responsive, futuristic theme
Features	Property CRUD, user/role management, logs, background checks

---

5. API Endpoints (Backend)

Method	Endpoint	Description	Auth Required	Roles
POST	/api/auth/login	User login	No	-
POST	/api/auth/register	User registration	No	-
GET	/api/properties	List properties	Yes	Admin,Agent
POST	/api/properties	Create property	Yes	Admin,Agent
PUT	/api/properties/{id}	Update property	Yes	Admin,Agent
DELETE	/api/properties/{id}	Delete property	Yes	Admin,Agent
GET	/api/users	List users	Yes	Admin
PUT	/api/users/{id}/role	Change user role	Yes	Admin

GET	/api/logs	View access logs	Yes	Admin,Agent
POST	/api/background-checks	Trigger background check	Yes	Admin,Agent
GET	/api/background-checks/{id}	Check status/result	Yes	Admin,Agent

---

6. Data Model Overview

```
User: id, username, password, email, role, created_at
Property: id, title, address, description, owner_id, status, created_at
AccessLog: id, user_id, action, entity, entity_id, timestamp, details
BackgroundCheck: id, target_type, target_id, status, result, requested_by, creat
```

---

7. Security & Compliance

- All endpoints secured via JWT; role-based access enforced
- Passwords hashed (BCrypt)
- Sensitive actions logged via AOP
- Input validation and sanitization throughout
- CORS configured for frontend-backend communication

---

8. Deployment & Operations

Aspect	Specification
Docker	Dockerfile for backend/frontend, docker-compose
Environments	Dev, Staging, Production via env variables
Open Source	MIT License, public GitHub repository
CI/CD	GitHub Actions for build/test/deploy

---

9. Non-Functional Requirements

Requirement	Specification
Performance	<300ms API response for 95% of requests
Scalability	Stateless backend, scalable via containers
Availability	99.5% uptime target
Usability	Intuitive, modern UI
Documentation	API docs (Swagger/OpenAPI), user guides

---

10. Milestones & Timeline

Milestone	Target Date
Requirements Finalized	Week 1
Backend MVP	Week 3
Frontend MVP	Week 5
Async/Logging Complete	Week 6
Dockerization	Week 7
Testing & QA	Week 8
Public Release	Week 9

---

11. Open Issues & Risks

- Async background check integration with external services
- Ensuring robust JWT security and token revocation
- Data migration and backup strategies

---

12. Acceptance Criteria

- All core features implemented and tested
- Security and logging requirements met
- Dockerized deployment works end-to-end
- Documentation complete and up-to-date
- All critical bugs resolved

End of Document