DevPortal

Centralized Credentials & Environment Management Portal

Cursor PRD · Implementation Spec · v1.0

Stack	Java (Spring Boot) · React · PostgreSQL · JWT
Target Users	All developers (Read), Admins (Read + Write)
Goal	Eliminate dependency on senior devs for onboarding credentials
Version	1.0 — Initial Release

1. Project Overview

DevPortal is an internal web application that acts as a single source of truth for all project credentials, environment configurations, API keys, database URLs, and other secrets. A new joiner can log in, navigate to their project and environment, and find everything they need without pinging a senior developer.

The portal supports multiple projects (e.g., PaymentService, AuthService, DataPipeline) and multiple environments per project (dev, staging, UAT, production). Access to credentials is role-gated: regular developers can view, while admins can create, update, and delete.

2. Goals & Non-Goals

2.1 Goals

- Provide a self-service portal for developers to access project credentials without senior involvement
- Support multi-project, multi-environment credential organization
- Role-based access: Admin (CRUD) vs Developer (Read-Only)
- Audit log — track who viewed or modified which credential and when
- Credential masking — secrets are hidden by default, revealed on explicit click
- Admin dashboard to manage users, projects, environments, and credentials

2.2 Non-Goals (v1.0)

- Secret rotation / auto-sync with AWS Secrets Manager or Vault (future)

- Fine-grained per-user-per-project access control (all devs share read access in v1)
- Mobile app

3. User Roles & Permissions

Role	Login	Credentials	Admin Actions
Developer	✓	View (masked by default)	✗
Admin	✓	View + Create + Edit + Delete	✓Full access

4. Feature Specifications

4.1 Authentication

- JWT-based login (email + password)
- Spring Security for auth filter chain
- Tokens stored in httpOnly cookies or localStorage (configurable)
- Password hashing with BCrypt
- Token expiry: 8 hours access token, 7 day refresh token

4.2 Project Management (Admin Only)

- Create / Edit / Archive projects
- Each project has: name, description, team, created_at, status
- Projects are listed in sidebar for all users

4.3 Environment Management (Admin Only)

- Each project can have N environments (dev, staging, UAT, prod, etc.)
- Environments are color-coded (dev=green, staging=yellow, prod=red)
- Admin can add/remove environments per project

4.4 Credential Management

- Each credential entry: key (e.g., DB_URL), value, description, type (secret/non-secret), last_updated, updated_by
- Secret values are masked (***) by default — click the eye icon to reveal
- Copy-to-clipboard button on each credential
- Admin can add / edit / delete credentials
- Bulk import via CSV (Admin only)

4.5 Audit Log

- Every view, create, update, delete action is logged
- Log entry: user, action, project, environment, credential_key, timestamp, ip_address
- Admin can view full audit log with filters (by user, project, date range)

4.6 User Management (Admin Only)

- Invite users by email
- Assign role: DEVELOPER or ADMIN
- Deactivate/reactivate accounts

4.7 Search

- Global search bar — search credentials by key name across all projects/envs
- Filter by project, environment, type

5. Data Models

5.1 Core Entities

Entity	Key Fields	Relationships
User	id, name, email, password_hash, role, is_active	Has many AuditLogs
Project	id, name, description, team, status	Has many Environments
Environment	id, project_id, name, color_code	Belongs to Project, Has many Credentials
Credential	id, env_id, key, value (encrypted), description, type, updated_by, updated_at	Belongs to Environment
AuditLog	id, user_id, action, project_id, env_id, credential_key, ip, timestamp	Belongs to User

6. REST API Endpoints

6.1 Auth

Method	Endpoint	Auth	Description
POST	/api/auth/login	Public	Login, returns JWT
POST	/api/auth/refresh	Public	Refresh access token
POST	/api/auth/logout	Bearer	Invalidate token

6.2 Projects

Method	Endpoint	Auth	Description
GET	/api/projects	Any	List all projects
POST	/api/projects	Admin	Create project
PUT	/api/projects/{id}	Admin	Update project
DELETE	/api/projects/{id}	Admin	Archive project

6.3 Environments

Method	Endpoint	Auth	Description
GET	/api/projects/{pid}/envs	Any	List environments
POST	/api/projects/{pid}/envs	Admin	Create environment
PUT	/api/projects/{pid}/envs/{id}	Admin	Update environment
DELETE	/api/projects/{pid}/envs/{id}	Admin	Delete environment

6.4 Credentials

Method	Endpoint	Auth	Description
GET	/api/envs/{eid}/credentials	Any	List all credentials (values masked)
GET	/api/envs/{eid}/credentials/{id}/reveal	Any	Reveal single credential value (logs access)
POST	/api/envs/{eid}/credentials	Admin	Add credential
PUT	/api/envs/{eid}/credentials/{id}	Admin	Update credential
DELETE	/api/envs/{eid}/credentials/{id}	Admin	Delete credential

Method	Endpoint	Auth	Description
POST	/api/envs/{eid}/credentials/import	Admin	Bulk import via CSV

6.5 Admin — Users & Audit

Method	Endpoint	Auth	Description
GET	/api/admin/users	Admin	List users
POST	/api/admin/users/invite	Admin	Invite new user
PUT	/api/admin/users/{id}	Admin	Update role/status
GET	/api/admin/audit-logs	Admin	View audit logs (filterable)

7. Frontend Structure (React)

7.1 Pages

Route	Description
/login	Login page
/dashboard	Home — project cards grid
/projects/:id	Project detail — list of environments
/projects/:id/env/:envId	Credential list for a project+env
/admin/users	User management
/admin/projects	Project/env management
/admin/audit-logs	Audit log viewer
/profile	User profile + password change

7.2 Component Architecture

- Layout: Sidebar (project nav) + TopBar (user avatar, search) + Main Content
- CredentialCard: key | masked-value | eye-icon | copy-icon | edit/delete (admin)
- EnvTabs: tab strip for switching between environments within a project
- AdminGuard: HOC that wraps admin-only routes
- useAuth hook: manages JWT, user role, login/logout
- useCredentials hook: fetch, reveal, copy logic

7.3 State Management

- React Context + useReducer for auth state (no external library needed for v1)
- React Query (TanStack Query) for all server data — caching, refetching, loading states
- Axios with interceptors for JWT attach + 401 handling

8. Backend Structure (Spring Boot)

8.1 Package Structure

```
com.devportal
  ├── config/          — SecurityConfig, JwtConfig, CorsConfig
  ├── controller/      — AuthController, ProjectController, EnvController,
CredentialController, AdminController
  ├── service/         — AuthService, ProjectService, CredentialService,
AuditService, EncryptionService
  ├── repository/      — JPA repositories for each entity
  ├── model/           — JPA entity classes
  ├── dto/             — Request/Response DTOs
  ├── security/        — JwtFilter, UserDetailsServiceImpl
  └── exception/       — GlobalExceptionHandler, custom exceptions
```

8.2 Key Technical Decisions

Concern	Decision
Credential encryption	AES-256-GCM via Java Crypto, key stored in env var / Vault
Auth	Spring Security + JJWT library
DB	PostgreSQL with Flyway migrations
ORM	Spring Data JPA / Hibernate
Validation	Jakarta Bean Validation (@Valid) on all DTOs
Error handling	GlobalExceptionHandler (@RestControllerAdvice)
Logging	SLF4J + Logback, MDC for request tracing
Testing	JUnit 5 + Mockito + Testcontainers for DB tests

9. Security Considerations

- All credential values encrypted at rest with AES-256-GCM before DB storage
- Encryption key injected via environment variable — never hardcoded

- HTTPS enforced in all environments (TLS termination at reverse proxy)
- CSRF protection enabled for cookie-based auth flows
- Rate limiting on /api/auth/login to prevent brute force (e.g., Bucket4j)
- Audit log every reveal action — builds accountability culture
- Admin endpoints protected by @PreAuthorize('hasRole(ADMIN)')
- Secrets never returned in list APIs — only on explicit /reveal call

10. Database Schema (PostgreSQL)

Table	Primary Key	Foreign Keys	Notable Columns
users	id (UUID)	—	email (unique), role, is_active, password_hash
projects	id (UUID)	—	name (unique), team, status
environments	id (UUID)	project_id → projects	name, color_code
credentials	id (UUID)	env_id → environments, updated_by → users	key, value_encrypted, type, description
audit_logs	id (UUID)	user_id → users, project_id, env_id	action, credential_key, ip_address, created_at

11. Repository & Folder Structure

Frontend (React)

```
devportal-frontend/
  ├── public/
  ├── src/
  │    ├── api/           — axios instances, all API call functions
  │    ├── components/    — shared UI components
  │    ├── pages/         — one folder per route
  │    ├── hooks/         — useAuth, useCredentials, useProjects
  │    ├── context/       — AuthContext
  │    ├── utils/         — helpers, constants
  │    └── App.jsx        — routes
  ├── .env.example
  └── package.json
```

Backend (Spring Boot)

```
devportal-backend/
```

```
├── src/main/java/com/devportal/
│   ├── config/
│   ├── controller/
│   ├── service/
│   ├── repository/
│   ├── model/
│   ├── dto/
│   ├── security/
│   └── exception/
├── src/main/resources/
│   ├── application.yml
│   └── db/migration/  — Flyway SQL scripts
├── src/test/
└── pom.xml
```

12. Cursor Implementation Prompts

Use these prompts directly in Cursor to scaffold the application section by section.

Prompt 1 — Backend Bootstrap

> *Create a Spring Boot 3 project with Spring Security, Spring Data JPA, PostgreSQL, Flyway, and JJWT. Set up JWT auth with login/refresh endpoints, a JwtFilter, and a UserDetailsService backed by a 'users' table with id, email, password_hash, role (DEVELOPER/ADMIN), is_active columns. Include GlobalExceptionHandler and return consistent error JSON {code, message, timestamp}.*

Prompt 2 — Credential Encryption Service

> *Create an EncryptionService in Spring Boot that encrypts and decrypts strings using AES-256-GCM. The 32-byte key is injected from an environment variable ENCRYPTION_KEY. Provide encrypt(String plaintext): String and decrypt(String ciphertext): String. Store the IV prepended to the ciphertext in Base64.*

Prompt 3 — Project/Env/Credential CRUD

> *Create JPA entities Project, Environment, Credential. Create REST controllers with full CRUD. Credential GET list returns values as '***'. A separate GET /credentials/{id}/reveal decrypts and returns the value, and writes an AuditLog entry. Admin-only endpoints use @PreAuthorize('hasRole(ADMIN)').*

Prompt 4 — React App Scaffold

> *Create a React app with React Router v6, TanStack Query, and Axios. Set up AuthContext with login/logout/role. Create a PrivateRoute and AdminRoute. Build a sidebar layout with project list. Add pages: Login, Dashboard (project cards), ProjectDetail (env tabs +*

> *credential list). Each credential row shows: key, masked value with eye toggle (calls /reveal), copy-to-clipboard button, and admin edit/delete.*

Prompt 5 — Admin Panel

> *Build an Admin section in React with three pages: User Management (invite user by email, set role, deactivate), Project/Environment Management (CRUD for projects and their environments), and Audit Log viewer (table with filters for user, project, date range — fetched from /api/admin/audit-logs).*

13. Development Milestones

Sprint	Deliverables	Duration
Sprint 1	Spring Boot setup, DB schema, JWT auth, User model	1 week
Sprint 2	Project + Env + Credential CRUD, EncryptionService, AuditLog	1 week
Sprint 3	React scaffold, Auth flow, Dashboard, Project/Env pages	1 week
Sprint 4	Credential reveal/copy/mask UI, Admin panel, User mgmt	1 week
Sprint 5	Audit log UI, Search, CSV import, Testing, Polish	1 week