# TQS: Product specification report

*José Rubem Lins de Aquino Neto [101092]*
v2026-01-29

## Contents

# 1 Introduction

## 1.1 Overview of the project

This project was developed in the context of the TQS (Testes e Qualidade de Software / Software Testing and Quality) course at Universidade de Aveiro. The main objective is to apply software quality assurance practices, including comprehensive testing strategies (unit, integration, and end-to-end tests), continuous integration/continuous deployment (CI/CD), static code analysis, and test-driven development principles.UA Volunteering Platform is a digital volunteering marketplace designed specifically for the Universidade de Aveiro (UA) community. The platform connects volunteering opportunities with students, faculty, and staff, enabling them to discover, apply for, and participate in meaningful volunteer activities across the university and surrounding community.The solution features a points-based rewards system that incentivizes volunteer participation. Volunteers earn points upon completing activities, which can be redeemed for rewards such as UA services, partner vouchers, merchandise, and certificates. This gamification approach aims to increase engagement and recognition of volunteer contributions within the academic community.Key features:

- Browse and search volunteering opportunities with advanced filtering
- Apply to opportunities and track application status
- Opportunity management for promoters/organizations
- Points accumulation and redemption for rewards
- User profile management with skills tracking
- Administrative dashboard for user and reward management

## 1.2 Project limitations and know issues

Implemented features:

- User authentication and authorization with JWT
- Opportunity CRUD with status lifecycle (Draft → Open → In Progress → Completed)
- Application workflow (Pending → Approved/Rejected → Completed)
- Rewards catalog and management
- Skills-based matching between volunteers and opportunities
- Admin panel for user role management and rewards administration

Known limitations:

- Redemption flow: The reward redemption feature (spending points to claim rewards) is not fully implemented in the frontend
- Partner management: Partners entity exists but the full partner portal is not yet available
- Notifications: No email or push notification system for application status updates
- Image uploads: Opportunities and rewards do not support image attachments
- Reporting/Analytics: No dashboard for statistics on volunteer hours, participation rates, etc.
- Mobile responsiveness: While using Tailwind CSS, mobile optimization was not a primary focus

## 1.3 References and resources

Key Libraries and Frameworks:

Backend:

- Spring Boot 4.0 - REST API development with Java 25
- PostgreSQL 16 - Persistent data storage

- Flyway - Version-controller database migrations
- Spring Security + JWT - Stateless authentication
- SpringDoc OpenAPI (Swagger) - Interactive API documentation

Frontend:

- React 19 + TypeScript - Single-page application
- Vite 7 - Fast frontend bundling
- TanStack Router - Type-safe client-side routing
- Tailwind CSS 4 - Utility-first CSS framework

Testing

- JUnit 5, Mockito, REST Assured - Backend unit and integration testing
- Cucumber - Behavior-driven development (BDD) testing
- Vitest - Frontend unit testing
- Playwright - End-to-end testing

Devops:

- Github Actions - Automated CI/CD pipelines
- SonarCloud - Static code analysis
- JaCoCo - Code coverage reporting
- Docker, Docker Compose - Containerization for development and deployment

AI Tools:

- Cursor + Claude 4.5 Opus High

## 2 Product concept and requirements

### 2.1 Vision statement

Problem Statement:

Universidade de Aveiro has a vibrant community with numerous volunteering opportunities across faculties, student associations, and partner organizations. However, there is no centralized platform to discover these opportunities, leading to underutilization of volunteer potential and limited recognition for participants.Solution:UA Volunteering Platform serves as a centralized marketplace where:

- Organizations/Promoters can publish and manage volunteering opportunities
- Volunteers can discover, apply, and participate in activities matching their skills and interests
- Administrators can oversee the platform, manage users, and configure rewards

Key Value Propositions:

1. Discoverability - Centralized catalog of opportunities with filtering by skills, dates, and points
1. Engagement - Gamified experience with points and rewards to encourage participation
1. Recognition - Volunteers build a portfolio of completed activities and earned skills
1. Efficiency - Streamlined application process with status tracking for both volunteers and promoters

Similar Products:

- Volunteer Match (volunteermatch.org) - General volunteer matching platform
- BVMS (Better Impact Volunteer Management) - Enterprise volunteer management
- Golden (golden.com) - Campus engagement platform

Our solution differs by being specifically tailored for the UA academic community with a focus on skill development and point-based incentives.

## 2.2 Personas and Scenarios

Persona 1: Maria Santos - The Student Volunteer

Age: 21 yo
Role: 3rd-year Computer Science student at DETI
Goals: Build CV, develop soft skills, give back to community
Tech Comfort: High - uses mobile apps daily
Motivations: Certificate for CV, meeting new people, learning opportunities

**Scenario 1.1 - Discovering and applying:**

Maria has a free weekend and wants to find a volunteering opportunity related to technology. She opens the UA Volunteering Platform, filters opportunities by her skills (Programming, Communication) and available dates. She finds an event teaching coding to high school students, reads the description, and applies with a short message about her experience. Within two days, she receives approval and prepares for the event.

**Scenario 1.2 - Discovering and applying:**

After completing the coding workshop, Maria's application is marked as completed by the promoter. She receives 50 points in her account. Over the semester, she accumulates 200 points and browses the rewards catalog. She redeems a "Campus Coffee Voucher" for 80 points from a local partner café.

Persona 2: Professor João Almeida - The Opportunity Promoter

Age: 45 yo
Role: Associate Professor and Director of student outreach program
Goals: Find reliable volunteers for research/outreach events
Tech Comfort: Medium - prefers simple interfaces
Motivations: Difficulty finding committed volunteers, managing applications via email

**Scenario 2.1 - Creating an opportunity:**

Prof. Almeida is organizing a science fair for local schools. He logs into the platform as a Promoter, creates a new opportunity specifying the title, description, required skills (Communication, Science), location, dates, maximum volunteers (15), and points reward (75 points). He publishes it and the opportunity becomes visible to all volunteers.

**Scenario 2.2 - Managing applications:**

Within a week, 20 students apply. Prof. Almeida reviews applications, checking volunteer profiles and their previous experience. He approves 15 candidates and politely rejects others. After the event, he marks successful participants as "completed," automatically awarding them points.

Persona 3: Ana Rodrigues - The Platform Administrator

Age: 35 yo

Role: UA Staff - Student Affairs Office
Goals: Ensure platform integrity, manage rewards catalog, handle user issues
Tech Comfort: High - manages multiples systems

**Scenario 3.1 - Managing users and rewards:**

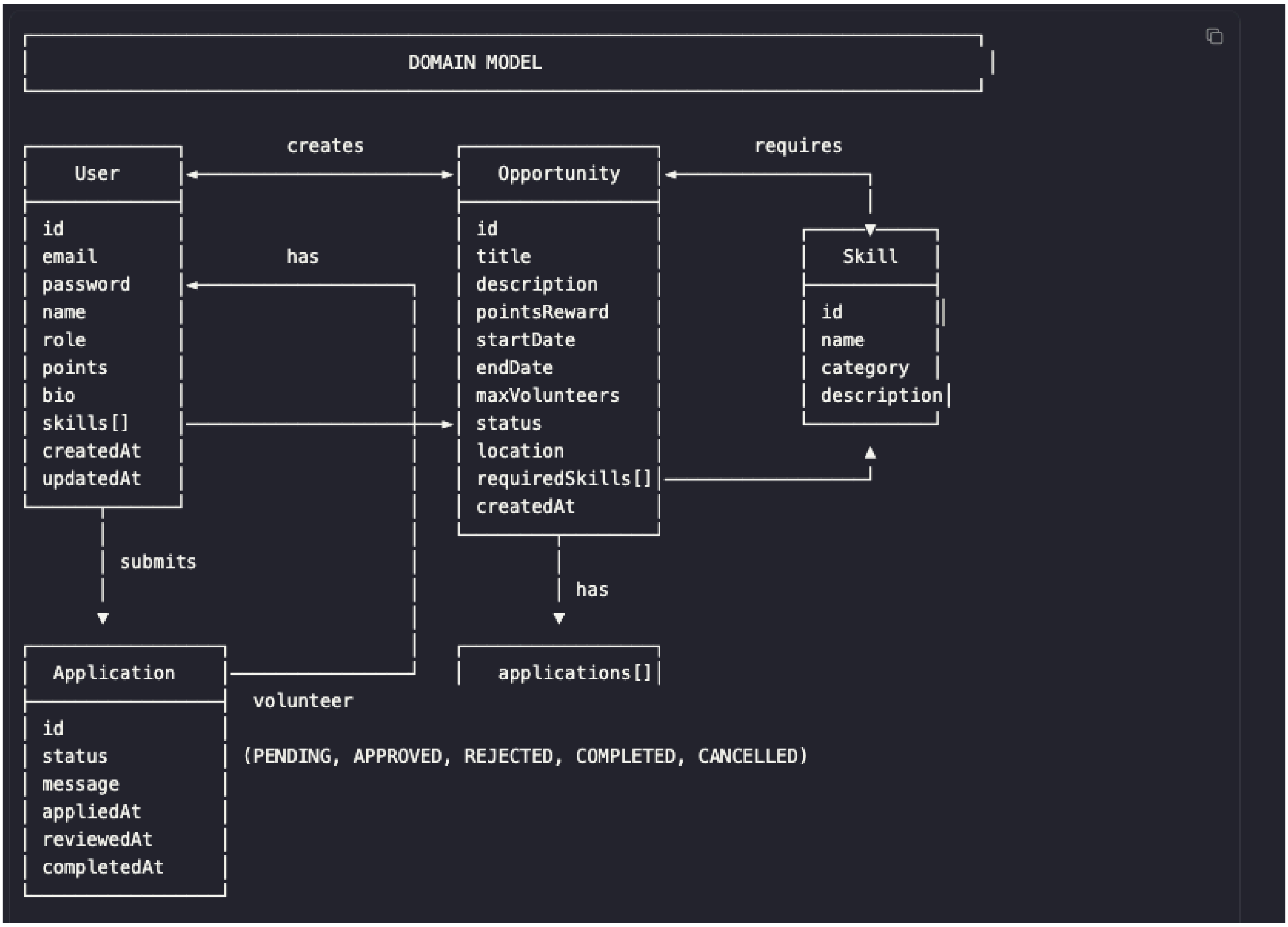
Ana notices a user account that needs role elevation from Volunteer to Promoter. She accesses the admin panel, searches for the user, and updates their role. She also adds new rewards to the catalog from a new partner café that joined the program, setting the point cost and availability dates.

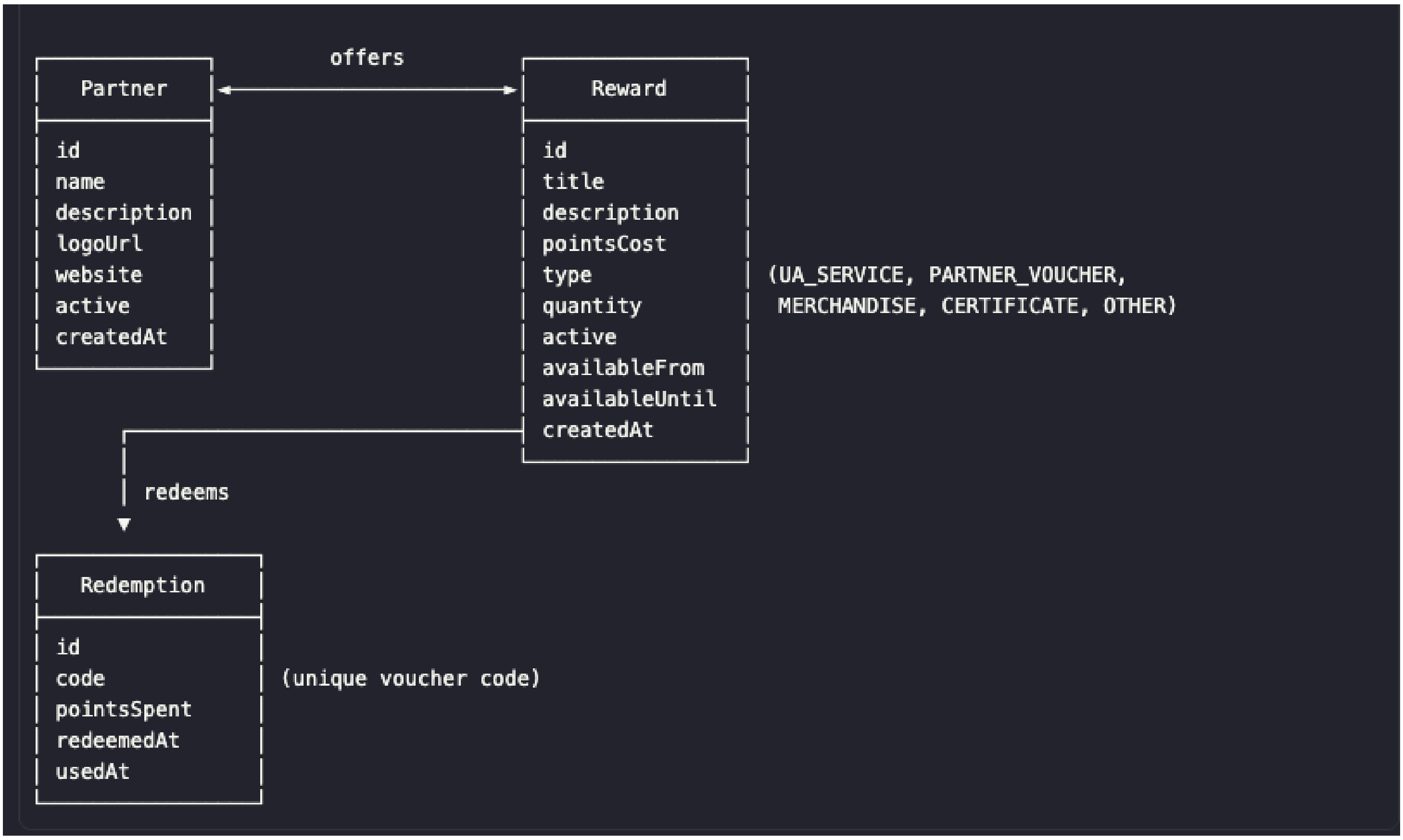
## 2.3 Project epics and priorities

The project was developed organized by the following epics:

- User Authentication: User registration, login, and session management
- User profile: Profile and skills management
- Opportunity Management: Core opportunity lifecycle
- Application Workflow: Volunteer application process
- User profile: Profile and skills management
- Rewards System: Points and rewards management
- Admin Dashboard: Platform administration
- Partner Portal: Partner-specific features

## 3 Domain Model





```
┌──────────────┐       offers         ┌─────────────────┐
│   Partner    │◄────────────────────►│     Reward      │
├──────────────┤                      ├─────────────────┤
│ id           │                      │ id              │
│ name         │                      │ title           │
│ description  │                      │ description     │
│ logoUrl      │                      │ pointsCost      │
│ website      │                      │ type            │ (UA_SERVICE, PARTNER_VOUCHER,
│ active       │                      │ quantity        │  MERCHANDISE, CERTIFICATE, OTHER)
│ createdAt    │                      │ active          │
└──────────────┘                      │ availableFrom   │
                                      │ availableUntil  │
       ┌──────────────────────────────│ createdAt       │
       │                              └─────────────────┘
       │ redeems
       ▼
┌─────────────────┐
│   Redemption    │
├─────────────────┤
│ id              │
│ code            │ (unique voucher code)
│ pointsSpent     │
│ redeemedAt      │
│ usedAt          │
└─────────────────┘
```

Key enumerations

- UserRole: VOLUNTEER, PROMOTER, PARTNER, ADMIN
- OpportunityStatus: DRAFT, OPEN, FULL, IN_PROGRESS, COMPLETED, CANCELLED
- ApplicationStatus: PENDING, APPROVED, REJECTED, COMPLETED, CANCELLED
- RewardType: UA_SERVICE, PARTNER:VOUCHER, MERCHANDISE, CERTIFICATE, OTHER
- SkillCategory: TECHNICAL, COMMUNICATION, ORGANIZATIONAL, CREATIVE, OTHER

# 4 Architecture notebook

## 4.1 Key requirements and constraints

Functional Requirements:

- RESTFUL API for all operations

- Secure authentications with JWT tokens
- Role-based access control
- Paginated and filterable data listings
- Real-time status tracking for applications

Non-Functional Requirements:

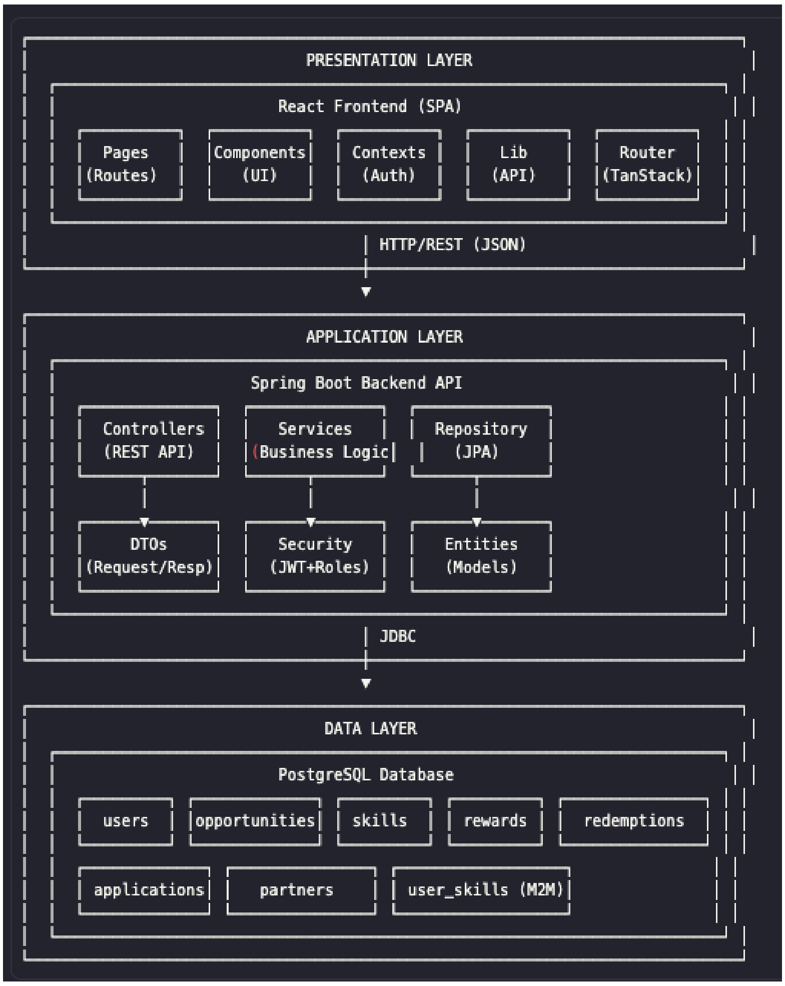
- Testability: Minimum 80% code coverage
- Maintainability: Clean architecture with separation of concerns
- Portability: Containerized deployment with Docker
- Security: Encrypted passwords, secure token handling
- Performance: Paginated queries to handle large datasets

Techinical Constraints:

- Must use Java (Spring Boot) for backend
- Must implement comprehensive testing (unit, integration, E2E, BDD)
- Must use CI/CD with Github Actions
- Must integrate SonarCloud for code quality analysis

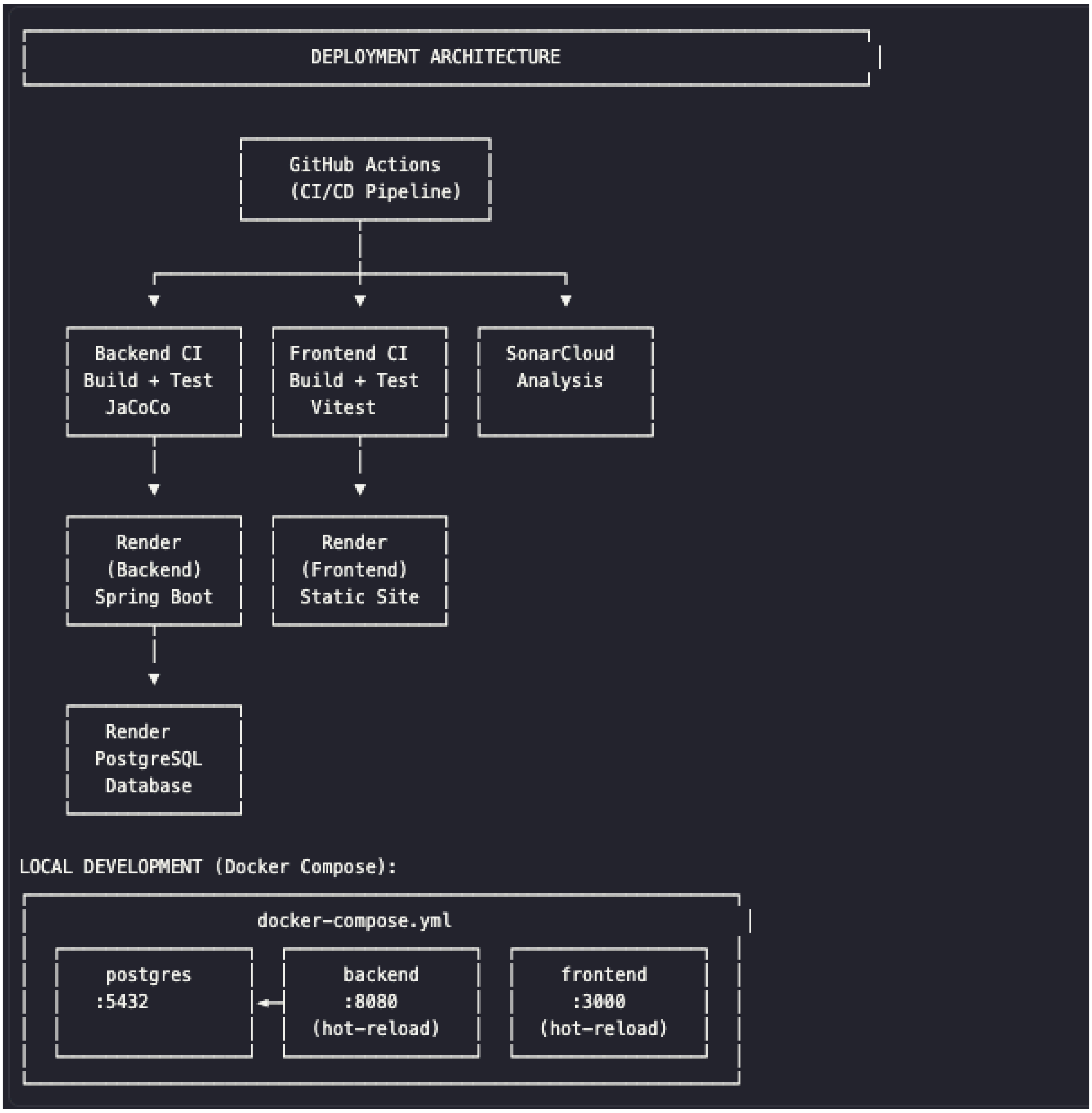
### 4.2 Architecture view

The system follows a three-tier architecture with clear separation:



```
┌─────────────────────────────────────────────────────────────────┐
│                       PRESENTATION LAYER                        │
│  ┌───────────────────────────────────────────────────────────┐  │
│  │                    React Frontend (SPA)                   │  │
│  │  ┌─────────┐  ┌─────────┐  ┌─────────┐  ┌─────────┐  ┌─────────┐  │  │
│  │  │  Pages  │  │Components│  │ Contexts │  │   Lib   │  │ Router  │  │  │
│  │  │(Routes) │  │  (UI)   │  │ (Auth)  │  │  (API)  │  │(TanStack)│  │  │
│  │  └─────────┘  └─────────┘  └─────────┘  └─────────┘  └─────────┘  │  │
│  └───────────────────────────────────────────────────────────┘  │
│                              │ HTTP/REST (JSON)                 │
└──────────────────────────────┼──────────────────────────────────┘
                               ▼
┌─────────────────────────────────────────────────────────────────┐
│                        APPLICATION LAYER                        │
│  ┌───────────────────────────────────────────────────────────┐  │
│  │                  Spring Boot Backend API                  │  │
│  │  ┌────────────┐  ┌────────────┐  ┌────────────┐           │  │
│  │  │ Controllers │  │  Services   │  │ Repository  │           │  │
│  │  │ (REST API)  │  │(Business Logic│  │   (JPA)    │           │  │
│  │  └────────────┘  └────────────┘  └────────────┘           │  │
│  │        │              │              │                    │  │
│  │  ┌─────▼──────┐  ┌─────▼──────┐  ┌─────▼──────┐           │  │
│  │  │    DTOs     │  │  Security   │  │  Entities   │           │  │
│  │  │(Request/Resp)│  │ (JWT+Roles) │  │  (Models)   │           │  │
│  │  └────────────┘  └────────────┘  └────────────┘           │  │
│  └───────────────────────────────────────────────────────────┘  │
│                              │ JDBC                             │
└──────────────────────────────┼──────────────────────────────────┘
                               ▼
┌─────────────────────────────────────────────────────────────────┐
│                           DATA LAYER                            │
│  ┌───────────────────────────────────────────────────────────┐  │
│  │                    PostgreSQL Database                    │  │
│  │  ┌───────┐  ┌─────────────┐  ┌───────┐  ┌─────────┐  ┌────────────┐  │  │
│  │  │ users │  │opportunities│  │ skills │  │ rewards │  │ redemptions │  │  │
│  │  └───────┘  └─────────────┘  └───────┘  └─────────┘  └────────────┘  │  │
│  │  ┌────────────┐  ┌────────────┐  ┌────────────────┐               │  │
│  │  │applications│  │  partners   │  │ user_skills (M2M)│               │  │
│  │  └────────────┘  └────────────┘  └────────────────┘               │  │
│  └───────────────────────────────────────────────────────────┘  │
└─────────────────────────────────────────────────────────────────┘
```

### 4.3 Deployment view



DEPLOYMENT ARCHITECTURE

GitHub Actions
(CI/CD Pipeline)

Backend CI
Build + Test
JaCoCo

Frontend CI
Build + Test
Vitest

SonarCloud
Analysis

Render
(Backend)
Spring Boot

Render
(Frontend)
Static Site

Render
PostgreSQL
Database

LOCAL DEVELOPMENT (Docker Compose):

docker-compose.yml

postgres
:5432

backend
:8080
(hot-reload)

frontend
:3000
(hot-reload)

## 5 API for developers

The API follows REST conventions and is documented with OpenAPI/Swagger (available at /swagger-ui.html)
Base URL

- Development: http://localhost:8080/swagger-ui/index.html
- Production: https://volunteering-api.onrender.com/swagger-ui/index.html

r