10/3/2025

CACODEV – Congolese Association for Congo Development Management System

By

David Katembo

Fabrice Kadima

Pemphyle Nzuzi

Indiana University Southeast

CSCI B438: CSCI P445 Capstone Project I Design Fall

2025

Professor Ronald Finkbine, Ph.D.

Breakdown of Contributions

David Katembo: Java (Spring Boot) backend development, database design, API implementation, authentication system. Introduction

Fabrice Kadima: Frontend (Angular), UI/UX, Stripe integration. Overall Description

Pemphyle Nzuzi: AWS deployment, Docker configuration, CI/CD pipelines, system security implementation. Specific Requirements

Software Requirements Specification (SRS)

CACODEV – Congolese Association for Congo Development Management System

1. Introduction

The CACODEV Management System is a comprehensive web-based solution developed for the *Congolese Association for Congo Development (CACODEV)*. The system aims to automate and streamline the organization's operations, including membership management, event scheduling, donation tracking, communication, and financial reporting.

By integrating modern technologies such as Spring Boot for backend development, Angular for frontend design, and Stripe API for secure payment processing, the platform ensures efficiency, transparency, and scalability. It provides role-based access for members, donors, and administrators, enabling secure interactions within a unified digital environment. The CACODEV Management System supports the association's mission by enhancing organizational efficiency, improving communication, and promoting long-term sustainability.

1) Purpose

The CACODEV Management System is a centralized platform designed to streamline the operations of the Congolese Association for Congo Development. It enables efficient management of memberships, events, fundraising, donations, communication, and organizational structures.

2) Definitions

Term	Definition
Angular	A TypeScript-based framework for building dynamic web applications.
API	Application Programming Interface that allows different software systems to communicate.
Application	A software program designed to perform a set of related functions for users.
AWS	Amazon Web Services, a cloud computing platform used for hosting and deployment.
Azure	Microsoft's cloud computing platform for hosting, deployment, and services.
Board Member	An administrator with elevated access rights to manage operations.
CI/CD	Continuous Integration and Continuous Deployment processes for automated software updates.
Docker	A containerization platform for packaging applications and their dependencies.
Donor	An individual or organization that provides financial contributions.
GCP	Google Cloud Platform, a suite of cloud services for hosting and deployment.
GDPR	General Data Protection Regulation for safeguarding personal data and privacy.

HTTPS (TLS/SSL)	Secure communication protocol using encryption to protect data in transit.
Interface	The boundary through which users or systems interact with the CACODEV platform.
Member	A registered user with CACODEV.
PCI DSS	Payment Card Industry Data Security Standard ensuring secure payment processing.
PostgreSQL/MySQL	Relational database management systems are used to store and manage structured data.
React.js	A JavaScript library for building interactive frontend user interfaces.
Server	A system that provides data, services, or resources to other systems or clients over a network.
Spring Boot	A Java framework for building production-ready backend services.
Stakeholder	Any person or organization with an interest in the CACODEV system's development and outcomes.
Stripe API	Online payment processing system for contributions and donations.
UI	User Interface, the visual and interactive elements of a system.
UX	User Experience design that ensures intuitive interaction with the system.

User	An individual interacting with the CACODEV system, including members, donors, and administrators.

3) System Overview

The system is a web-based application with a Spring Boot backend and Angular frontend, deployed via Docker on cloud infrastructure (AWS/Azure/Google Cloud). It integrates Stripe for secure payments and includes role-based access management.

4) References

- Angular. (n.d.-a). Angular material. Angular Material. https://material.angular.dev/
- Angular. (n.d.-b). Angular/components: Component infrastructure and Material Design components for angular. GitHub. https://github.com/angular/components
- IEEE Software Requirements Specification Template. (n.d.). https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc
- jam01. (n.d.). SRS-Template/template.md at master · jam01/SRS-template. GitHub. https://github.com/jam01/SRS-Template/blob/master/template.md#14-references
- NgRx docs. (n.d.-a). https://ngrx.io/guide/store?utm
- NgRx docs. (n.d.-b). https://ngrx.io/docs
- Sheldonlinker. (2021, December 3). Appendix C: IEEE 830 template. Requirements Engineering.

https://press.rebus.community/requirementsengineering/back-matter/appendix-c-ieee-830-template/

- Spring security. Spring Security :: Spring Boot. (n.d.). https://docs.spring.io/spring-boot/reference/web/spring-security.html?utm
- Track a payment link. Stripe Documentation. (n.d.). https://docs.stripe.com/payment-links/url-parameters

2. Overall Description

1) Product Perspective

The CACODEV system replaces manual workflows with an integrated digital platform.

i. System Interfaces

- Payment Gateway: Stripe API
- Hosting: AWS/Azure/Google Cloud
- Authentication: Spring Security (JWT-based)

ii. User Interfaces

- Angular-based responsive UI with Material Design.
- Dashboards for members, donors, and administrators.

iii. Hardware interfaces

- Runs on standard devices (PCs, laptops, tablets, smartphones).
- Accessed via modern browsers (Chrome, Edge, Firefox, Safari and more).

iv. Software interfaces

- Spring Boot REST APIs for backend communication.
- PostgreSQL/MySQL relational database.

v. Communication interfaces

- Secure communication over HTTPS (TLS/SSL).
- Email notifications for reminders, donations, and announcements.

vi. Memory Constraints

- Cloud-hosted, scalable storage and memory allocation.

vii. Operations

- Deployed in Docker containers.
- CI/CD pipelines ensure automated updates and continuous delivery.

viii. Site Adaptation Requirements

- Optimized for cloud hosting environments (AWS, Azure, GCP).

2) Product functions

I. Membership Management

- Register and manage members.
- Assign board roles and track contributions.

II. Fundraising & Donations

- Create and manage fundraising campaigns.
- Process payments securely via Stripe.
- Track donor information and contributions.

III. Events & Meetings

- Organize and schedule events.
- Manage meeting minutes, motions, and attendance tracking.

IV. Communication

- Blog posts and discussions.
- Announcements and updates.
- Centralized contact management.

V. User Access

- Role-based access (Admin, Member, Donor).
- Secure login and authentication with tokens.

VI. Financial Tracking

- Automated membership billing.
- Generate reports on income and expenses.

3) User characteristics

- Members: Typically, adults of varied technical skill levels; interact with the system to update personal information, register for events, and view announcements. They are generally familiar with basic web navigation but may require clear UI guidance.
- Donors: May include individuals or organizations; access the platform primarily to make financial contributions and view donation history. They expect secure, reliable payment processing and clear receipts.
- Board Members/Administrators: Users with elevated privileges responsible for managing members, events, and finances. They require detailed dashboards, reporting tools, and administrative controls. These

users are usually comfortable with web-based applications and may require training in advanced features like event management and financial reporting.

- General Characteristics: All users require responsive web access via desktop and mobile devices, secure authentication, and a straightforward, intuitive interface. Accessibility and inclusivity should be considered to accommodate diverse skill levels and devices.

4) Constraints, assumptions and dependencies

- Constraints:
    - System must comply with PCI DSS for payment processing and GDPR for user data protection.
    - Payment processing via Stripe must be completed within 5 seconds.
    - Web applications must support up to 500 concurrent users without performance degradation.
    - Deployment must utilize Docker containers for environment consistency across cloud platforms.
- Assumptions:
    - Users have access to modern browsers and stable internet connections.
    - Members, donors, and administrators are willing to use a centralized digital platform.

- Cloud services (AWS, Azure, or GCP) will provide sufficient resources for scaling storage and compute needs.
- External APIs (Stripe, email services) will remain available and operational during system use.

- Dependencies:
  - Stripe API for secure payment transactions.
  - PostgreSQL/MySQL databases for persistent data storage.
  - Cloud hosting services (AWS, Azure, GCP) for deployment and uptime.
  - Spring Boot backend and Angular frontend frameworks for core system functionality.
  - CI/CD pipelines to maintain automated deployments and updates.

3. Specification Requirements

1) External interface requirements

- Stripe API integration for payments.
- REST APIs for data exchange between backend and frontend.

2) Functional requirements

- Users can register and manage profiles.
- Admins can create and manage events.
- Donors can contribute via Stripe.
- System maintains historical data of donations, events, and meetings.

3) Performance requirements

- Support up to 500 concurrent users.
- Payment processing must be completed within 5 seconds.

4) Design constraints

- Must comply with PCI DSS (payment card industry security).
- Must comply with GDPR for data protection.
- Standards Compliance

5) Logical database requirement

- Relational schema with referential integrity.
- Track historical contributions and membership activity.

6) Software System attributes

- Reliability: 99.5% uptime.
- Availability: 24/7 cloud-based hosting.
- Security: Encrypted communication, RBAC (role-based access control).
- Maintainability: Modular microservice architecture.
- Portability: Runs on major browsers and deployable across AWS, Azure, or GCP.

4. Other requirements

- Weekly backups of critical data.
- System monitoring for uptime and performance.

5. Key Personnel Information

- David: Development Team, Team Leader
- Fabrice: Development Team

- Pemphyle: Development Team