BOOKBAZAAR

Project Proposal

Viktor Pavlov
v.pavlov@student.fontys.nl

Project Overview

BookBazaar is an e-commerce web application enabling users to buy and sell books online. The application will include key features such as book browsing, secure login, cart management, order processing, and an admin interface for managing book inventory. It will adhere to software development best practices including modular design, testing strategies, CI/CD integration, and containerization.

Project Objectives

- Build a secure, intuitive online platform for book transactions.
- Implement robust backend services for product management, order processing, and user authentication.
- Establish and maintain a comprehensive CI/CD pipeline.
- Validate architecture through modular design, UML diagrams, performance testing, code analysis, and stakeholder feedback.
- Deploy using Docker containers to public cloud services.

Design and Architecture

The system architecture will follow SOLID principles, ensuring modularity, maintainability, and scalability. A detailed UML diagram will outline the following modules:

- **Frontend Module**: React-based components for user interactions.
- **Backend Module**: Java Spring Boot for business logic, including:
    - User Authentication Module
    - Book Management Module
    - Order Processing Module
- **Database Module**: MySQL for persistent data storage.
- **Admin Module**: Secure management of book inventory and user orders.
- **Testing Module**: Frameworks and scripts for multiple test types (unit, integration, E2E).

Testing Strategy (Testing Pyramid)

The project will adopt a structured testing approach following the testing pyramid:

- **Unit Tests**: Achieve approximately 80% coverage on business logic and critical backend components.
- **Integration Tests**: Ensure correct interactions between individual modules (already planned).
- **End-to-End (E2E) Tests**: Develop at least 2-3 automated E2E tests covering critical user journeys such as login, checkout, and inventory management.

User Acceptance Criteria

- As a user, I can browse a list of available books so that I can explore options.

- As a user, I can search for books by title, author, or category to find what I want quickly.

- As a user, I can view detailed information about a book including title, author, price, and description.

- As a user, I can register and log into my account so that I can manage my purchases.

- As a user, I can add books to a shopping cart and review or update the cart before checkout.

- As a user, I can complete the purchase of books via a checkout process and receive confirmation.

- As a user, I want to sign up for notifications when an out-of-stock book becomes available, so I can promptly purchase or borrow it.

- As a user, I want to submit a request for books not currently available in the catalog, so librarians can consider adding them to inventory.

- As an admin, I can log in to an admin panel to manage book listings (add/edit/delete).

- As an admin, I can view and manage customer orders.

- As a developer, I can use a RESTful API to connect frontend and backend functionalities.

- As a developer, I can run unit and integration tests to ensure application quality.

- As a developer, I can apply static code analysis with SonarQube to maintain code quality.

Tech Stack

- **Frontend**: React
- **Backend**: Java Spring Boot
- **Database**: MySQL
- **CI/CD**: GitHub Actions
- **Containerization**: Docker
- **Code Quality**: SonarQube

Timeline (6 Weeks)

- **Week 1**: Scope definition, requirements gathering, UML diagram, and architecture design.
- **Week 2**: Initial frontend/backend setup and basic routing.
- **Week 3**: Implement user login, shopping cart, and admin functionalities.
- **Week 4**: Develop unit, integration, and initial E2E tests; apply static analysis.
- **Week 5**: Establish CI/CD pipeline, Dockerize application, and begin cloud deployment.
- **Week 6**: Complete UAT, address feedback, finalize documentation.

Deliverables

- Live deployed e-commerce application
- Source code repository with structured CI/CD pipeline
- Docker configuration scripts
- Complete test documentation (unit/integration/E2E)
- Final comprehensive documentation including UML design diagrams, security assessment (OWASP), and UAT reports.