A Java Spring Boot Web Application

Mohammadreza Motallebi | 7029006 | Spring 2024 | UNIFI

Advanced Techniques and Tools for Software Development

User Management and Quiz System

Introduction

The **User Management and Quiz System** is a full-stack **Spring Boot** web application designed for educational platforms, allowing users to register, participate in quizzes, and track their progress. The project integrates **Test-Driven Development (TDD), CI/CD automation, Dockerized testing, and Static Code Analysis (SonarQube Cloud integration)** to ensure high software quality and maintainability.

Technologies and Tools Used

Backend Technologies

- **Spring Boot (v3.2.4)** – RESTful API development
- **Spring Data JPA** – Database interaction (**PostgreSQL**)
- **Spring Boot Actuator** – Application health monitoring

Frontend Technologies

- **Thymeleaf** – Java-based template engine
- **HTML, CSS** – Styling and interactivity

Testing & Quality Assurance

- **JUnit 5 & Mockito** – Unit testing
- **Selenium WebDriver** – End-to-end (E2E) testing
- **Testcontainers** – Database-driven integration testing
- **JaCoCo** – Enforced **100% code coverage**
- **SonarQube Cloud** – **Static Code Analysis & Technical Debt Monitoring**

Security & DevOps

- **Docker & Docker Compose** – Containerized deployment
- **Maven (Build Automation)** – Automated dependency and build management
- **GitHub Actions (CI/CD)** – Automated testing & deployment
- **Environment Variables (.env)** – Secure configuration management

---

System Architecture

The project follows a **layered architecture**:

1. **Controller Layer (API Endpoints)**
   - `UserController.java`, `QuizController.java`, `WebController.java`
2. **Service Layer (Business Logic)**
   - `UserService.java`, `QuizService.java`
3. **Repository Layer (Database Interaction)**
   - `UserRepository.java`, `QuizResultRepository.java`
4. **Persistence Layer**
   - **PostgreSQL, Spring Data JPA, init.sql** (Database initialization)
5. **Testing Layer**
   - **Unit Tests:** `UserServiceTest.java`
   - **Integration Tests:** `UserServiceIT.java`
   - **E2E Tests:** `CombinedE2ETest.java`

Key Features

User Management

- User registration and login
- User profile management (CRUD operations)

Quiz System

- Quiz participation with results tracking
- Results stored with timestamps for analytics

Testing Strategy

- **Unit Tests** – Validate individual methods
- **Integration Tests** – Validate end-to-end database interactions
- **E2E Tests** – Simulate real-world user interactions

Logging & Monitoring

- **SLF4J + Logback** – Structured logging
- **Spring Boot Actuator** – Real-time health monitoring

Database Schema & Entity Relationships

- **User (id, username, email, birthdate, etc.)**
  - **1:N → QuizResult** *(One user can have multiple quiz results)*
- **QuizResult (id, user_id, score, timestamp)**

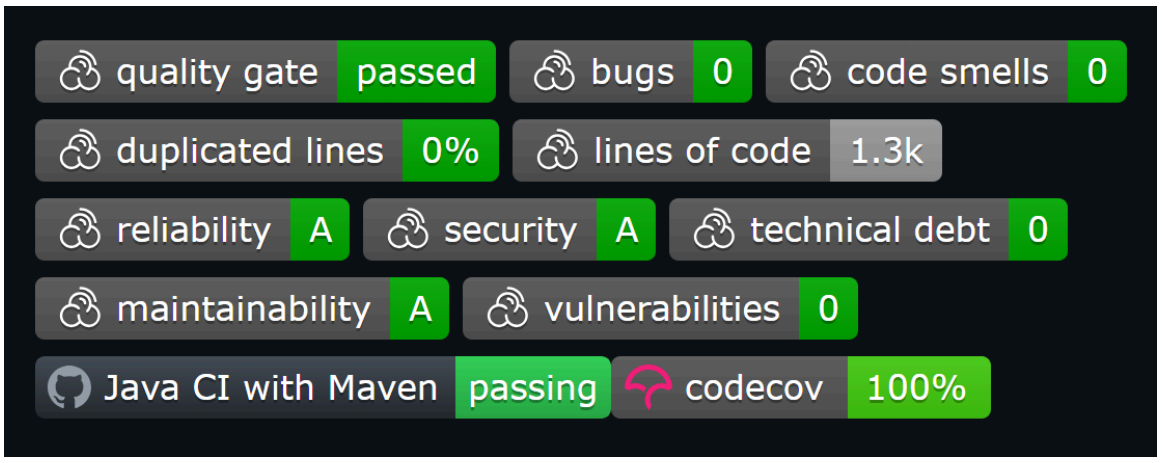
- **N:1 → User** *(Each quiz result belongs to one user)*

SonarQube Static Code Analysis

The **project is integrated with SonarQube Cloud**, enforcing high-quality coding standards.

SonarQube Analysis Summary

- **Lines of Code (LOC): 1.3k**
- **Quality Gate Status: ✅ Passed**
- **Code Coverage: 100%**
- **Duplications: 0.0%**
- **Security Issues: 0**
- **Maintainability Issues: 0**



SonarQube ensures **"0 Technical Debt"** and enforces **industry best practices**.

Deployment & Execution

Running Locally

1. Clone the repository.
2. Run Docker:

   ```
   docker-compose up --build
   ```

3. Application available at `http://localhost:8081`.

Automated Testing

- **Run unit tests**

  ```
  mvn test
  ```

- **Run integration tests**

  ```
  mvn verify
  ```

Continuous Integration

- **GitHub Actions + SonarQube Cloud** automates:
  - Build process
  - Static Code Analysis
  - Quality Gate Verification

Key Challenges and Solutions

Challenges Faced

1. **Maven & Test Execution Issues**
   - Incorrect execution of Integration and E2E tests using `mvn test` instead of `mvn verify`.
   - Missing `maven-failsafe-plugin` for proper test execution.
   - Test files were not structured correctly, leading to conflicts.
2. **Docker & Database Configuration Issues**
   - PostgreSQL service initialization failures.
   - Manual startup required for Docker and Spring Boot.
   - Spring Boot failed to detect PostgreSQL in Docker due to incorrect hostname resolution.
3. **CI/CD & Testing Issues**
   - **E2E tests were execution-dependent**, causing instability.
   - Missing dependencies in GitHub Actions pipeline, leading to CI failures.
   - `.gitignore` was not properly configured.
4. **Spring Boot & Configuration Issues**
   - **Redundant dependencies and misconfigured properties**.
   - **Inconsistent Java versions across local and CI/CD environments**.
5. **Selenium & WebDriver Issues**
   - ChromeDriver version mismatches caused Selenium failures.
   - Missing **Selenium dependencies** prevented Chrome automation.

Solutions Implemented

✅ 1. Fixed Maven & Test Execution

- Structured tests into separate directories.
- Configured `maven-surefire-plugin` for unit tests and `maven-failsafe-plugin` for integration & E2E tests.
- Cleaned `pom.xml` dependencies.

✅ 2. Fixed Docker & PostgreSQL Configuration

- Updated `docker-compose.yml` with proper initialization.
- Added **health checks** to prevent premature application startup.
- Used **Testcontainers** for database integration testing.

✅ 3. Automated CI/CD Testing

- Configured GitHub Actions to start Docker & PostgreSQL before tests.
- Ensured **proper test isolation** to prevent shared state conflicts.

✅ 4. Refactored Spring Boot Configurations

- Cleaned `application.properties` and standardized Java versions.

✅ 5. Fixed Selenium & WebDriver Issues

- Ensured **ChromeDriver version compatibility**.
- Added **dependencies**.

✅ 6. Final Validation & Deployment

- Successfully ran `mvn clean install` and `mvn verify`.
- Ensured **Docker, PostgreSQL, and Spring Boot work together seamlessly**.
- Application is **deployable, stable, and production-ready**.

Final Outcome

🚀 **The project is now fully functional, CI/CD-ready, and follows industry best practices!** 🎉