**Quizeria: Online Quiz Platform**

**by Naman Jain**

**1. Introduction**

* **Quizeria** is a web-based quiz management system built to simplify the process of conducting online quizzes in educational institutions.
* It uses **Spring Boot** and **Thymeleaf** to provide a clean, secure, and interactive environment for both administrators and students.
* The application features complete role-based access with a custom user interface using only HTML and CSS — no external frontend frameworks like Bootstrap are used.
* Designed for ease of demonstration, the system uses in-memory storage for data, avoiding external database configurations.

**2. Objectives**

* **Streamlined Quiz Management:**  
  Simplify the process for teachers and admins to create, manage, and organize quizzes efficiently.
* **Student Engagement and Feedback:**  
  Provide a simple platform for students to attempt quizzes, view results, and offer feedback.
* **Role-Based Security:**  
  Implement secure, role-based access to ensure admins and students have access to appropriate sections only.
* **Extensibility:**  
  The system is built to be easily extendable, making it ready for future features like database integration, analytics, or advanced reporting.

**3. Technologies Used**

* **Spring Boot:**  
  Acts as the backend framework for rapid development and dependency injection, with built-in web and security support.
* **Thymeleaf:**  
  Used as the templating engine for server-side rendering of dynamic HTML pages.
* **Spring Security:**  
  Handles user login, logout, and access control based on predefined roles (ADMIN and USER).
* **HTML & CSS (No UI Libraries):**  
  Custom styling ensures a lightweight and responsive frontend experience without dependency on libraries like Bootstrap.
* **Maven:**  
  Manages project dependencies and simplifies the build lifecycle.
* **In-Memory Storage:**  
  All data is stored in Java collections during runtime — suitable for demo or test environments.

**4. System Architecture**

* **Model-View-Controller (MVC):**  
  Quizeria follows the MVC pattern for clean separation of concerns between the backend logic and user interface.
* **Model Layer:**  
  Includes Java classes for entities like Quiz, Question, User, and Feedback.
* **View Layer:**  
  Built using Thymeleaf templates, this layer renders user-facing web pages with embedded logic.
* **Controller Layer:**  
  Manages HTTP requests, routes user actions, and handles access control logic using annotations and Spring Security rules.
* **Security Layer:**  
  Controllers restrict URLs and features based on roles (admin/student), ensuring protected access to critical actions.

**5. Features**

**For Administrators**

* **Dashboard Access:**  
  Secure admin dashboard with an overview of all available quizzes and their statistics.
* **Quiz Management:**  
  Ability to add, edit, or delete quizzes and questions using dynamic forms.
* **Feedback Viewing:**  
  Access student-submitted feedback to improve quiz content or user experience.

**For Students**

* **Quiz Participation:**  
  List of available quizzes, with real-time quiz attempt interface and result generation.
* **Score Display:**  
  Immediate display of quiz results with basic performance metrics.
* **Feedback Submission:**  
  Optional feedback form after each quiz to rate the quiz and suggest improvements.

**Common Features**

* **Custom Login/Logout:**  
  Personalized login experience with secure authentication and logout redirection.
* **Home Page Navigation:**  
  A landing page that allows users to choose their role and proceed accordingly.

**6. Project Structure**

* **Source Code Layout:** Organized under src/main/java/com/quizeria/, with sub-packages for:
  + controller/: Web layer handling HTTP requests.
  + model/: Data structures for business logic.
  + config/: Security and MVC configurations.
* **Templates Folder:** All Thymeleaf templates (HTML pages) are located in src/main/resources/templates/.
* **Static Resources:** CSS files and other static content are placed under src/main/resources/static/.
* **Configuration File:** application.properties includes basic settings like port, context paths, and security preferences.

**7. Security**

* **Spring Security Integration:**  
  Configured to provide in-memory users with encrypted credentials.
* **Roles and Access Control:**
  + ADMIN: Can access all quiz creation and management features.
  + USER: Limited to quiz participation and feedback.
* **CSRF Protection:**  
  Enabled by default, ensuring form submissions are protected from malicious cross-site attacks.
* **Secure Login/Logout Flow:**  
  Users are redirected to dashboards based on roles after authentication.

**8. How to Run the Application**

**Prerequisites:**

* **Java 17 or higher**
* **Maven 3.6+**

**Steps to Run:**

Open your terminal and type the following command:

cd quizeria

mvn spring-boot:run

**Accessing the Application:**

* Visit: http://localhost:8080/

**Demo Credentials:**

* **Admin**  
  Username: admin  
  Password: admin123
* **Student**  
  Username: student  
  Password: student123
* Note: Since the data is stored in memory, any added quizzes or changes will be lost once the server stops.

**9. Limitations and Future Enhancements**

**Limitations:**

* **No Persistent Storage:**  
  All data resets on restart due to in-memory design.
* **Limited User Management:**  
  No registration or profile management — only hardcoded user roles exist.
* **Basic Styling:**  
  Interface is minimalist, without responsive design or mobile-first layouts.

**Future Enhancements:**

* **Database Integration (e.g., MySQL, PostgreSQL):**  
  Allow persistent storage of quizzes, users, and results.
* **Advanced Analytics:**  
  Introduce quiz statistics, time tracking, and user performance charts.
* **User Registration:**  
  Self-sign-up and email verification systems for students and admins.
* **Responsive Design:**  
  Integration with modern UI libraries like Bootstrap or Tailwind for improved UX.
* **Email Notifications:**  
  Auto-send quiz results or reminders via email.
* **Password Recovery:**  
  Secure "Forgot Password" workflow with token-based verification.

**10. Conclusion**

* **Quizeria** demonstrates a secure, modular, and role-based online quiz system using pure Spring Boot and Thymeleaf without database or frontend libraries.
* The project showcases real-world implementation of user authentication, session control, and MVC design in a minimal setup.
* Its clean codebase and structured architecture offer a strong foundation for academic, demo, or future enterprise-level enhancements.