Online Survey G Feedback System

Submitted by:

Name : Johnson Kumar

U.I.D : 23bcs12654

Project Description

The *Online Survey & Feedback System* is a web-based application designed to create, manage, and analyse surveys efficiently. It enables administrators to build customized surveys, participants to provide responses through a user-friendly portal, and stakeholders to evaluate results using interactive dashboards. Built using *Spring Boot* for backend services, *React.js* for the frontend interface, and *MongoDB* for scalable data storage, the system provides a dynamic and secure platform for conducting surveys across academic, corporate, and research domains.

Key Features:

- Dynamic survey creation with multiple question types.
- Secure participant portal for submitting responses.
- Real-time aggregation and visualization of results.
- Role-based access for administrators, creators, and participants.

Project Distribution into Modules

The project is divided into five core modules to ensure clarity, scalability, and effective development:

1. Frontend – Survey Builder & Participant Portal (React.js)

- Survey Builder UI:
    - Create dynamic forms (MCQs, text fields, ratings, etc.).
    - Add, edit, and arrange questions.
    - Preview surveys before publishing.
- Participant Portal UI:
    - Display available surveys.

    - Provide responsive forms to submit answers.
    - Ensure a clean and mobile-friendly design.

2. Frontend – Result Dashboard & Admin Panel (React.js)

- Result Dashboard:
    - Show aggregated results in visual charts/graphs (Chart.js/Recharts).
    - Enable filtering by date, survey type, or user group.
    - Provide export options (CSV, Excel, PDF).
- Admin Panel:
    - Manage surveys (publish/unpublish/delete).
    - Control user roles (Admin, Creator, Participant).
    - Monitor participation rates and activity.

3. Backend & Database (Spring Boot + MongoDB)

- API Development (Spring Boot):
    - CRUD APIs for surveys and responses.
    - Authentication APIs (login, signup).
    - Secure communication with JWT tokens.
- Database (MongoDB):
    - Store surveys in JSON format (flexible schema).
    - Store participant responses and link them to surveys.
    - Optimize queries for fast analytics.
- Integration:
    - Connect backend APIs with frontend React components.
    - Deploy backend and database on cloud platforms (Render, MongoDB Atlas).

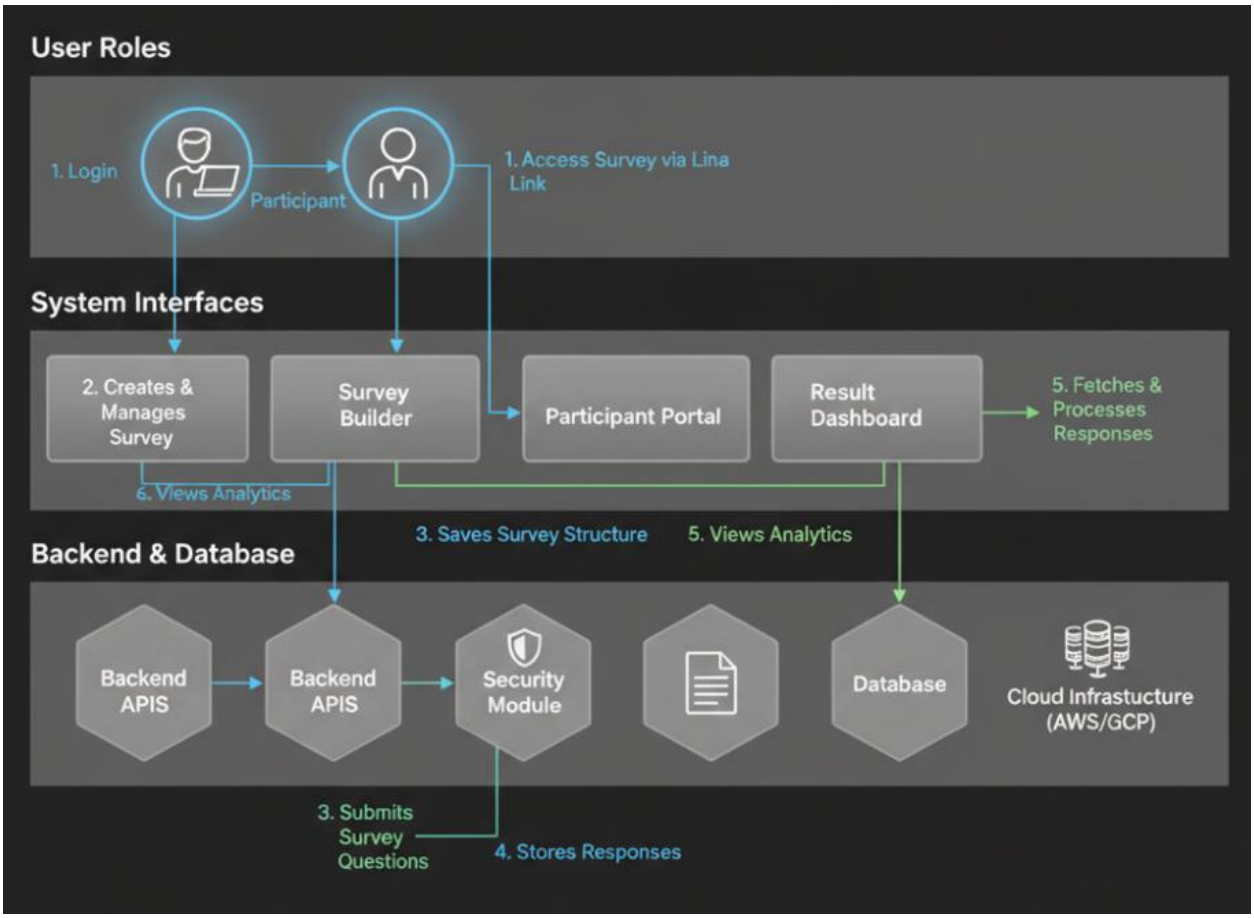
4. Security & Authentication Module

- Implement JWT-based authentication (login sessions).
- Role-based access control (Admin, Creator, Participant).
- Secure API endpoints with Spring Security.

- Input sanitization and validation to prevent attacks (e.g., SQL/NoSQL injection, XSS).

5. Integration & Deployment Module

- Connect frontend (React) with backend (Spring Boot APIs).
- Integrate backend with MongoDB Atlas (cloud-hosted DB).
- Add environment configs for development/production.
- Deploy application on platforms like Render / Vercel / Netlify + Heroku/Spring Boot server.



Report on Module 4: Security and Deployment

Security & Authentication Module

Overview:

The Security & Authentication module is the foundation of the system's protection architecture, ensuring that only authorized users can access specific features while safeguarding user data and communications. This module leverages Spring Security, JWT (JSON Web Tokens), and robust input validation mechanisms to build a secure and trustworthy environment for all system users.

JWT-Based Authentication:

Authentication in the Online Survey & Feedback System is powered by JWT tokens, which provide a stateless, scalable approach to managing user sessions. JWTs contain digitally signed payloads with encoded user information and roles.

Authentication Workflow:

1. A user submits login credentials (username/email and password).
2. The backend validates credentials against stored hashed passwords (using BCrypt).
3. Upon successful authentication, the server generates a JWT token signed with a secret key.
4. This token is sent back to the client and stored securely (in browser localStorage or sessionStorage).
5. Each subsequent request to protected API routes must include this token in the Authorization header.

Token Validation & Expiry:

- Every token includes an expiry timestamp to prevent long-lived sessions.

- Invalid or expired tokens trigger an authentication exception and require re-login.
- Refresh tokens may be implemented for seamless session renewal.

Role-Based Access Control (RBAC):

The system enforces granular role-based access control to regulate user permissions and actions. Each role (Admin, Creator, Participant) has predefined authorities within the application.

Defined Roles and Privileges:

- Administrator:
  - Full access to all modules including survey creation, modification, deletion, and analytics.
  - Manages user roles and system configurations.
- Creator:
  - Authorized to create, edit, and manage their own surveys.
  - Can view and analyze responses specific to their surveys.
- Participant:
  - Restricted access; can only view and respond to surveys.
  - Cannot alter survey structures or analytics.

Implementation:

- Access is enforced via Spring Security annotations (e.g., @PreAuthorize and @Secured).
- Method-level and endpoint-level security are implemented to isolate privileges.
- Unauthorized access attempts return HTTP 403 (Forbidden) responses.

Securing API Endpoints with Spring Security:

Spring Security acts as the primary shield against unauthorized requests and external attacks. The system is configured with custom SecurityFilterChain and AuthenticationManager components.

Key Security Layers:

- JWT Filter: Intercepts every incoming request, extracts the token, and

verifies its validity before allowing access to protected endpoints.

- CSRF Protection: Although JWTs minimize CSRF risk, additional CSRF prevention mechanisms can be enabled for sensitive operations.
- Password Encoding: All passwords are hashed using BCryptPasswordEncoder to ensure they are never stored in plain text.

API Endpoint Security Rules:

- /api/auth/** → Open for login and signup.
- /api/surveys/** → Restricted to authenticated users.
- /api/admin/** → Restricted to Admin role only.

Input Sanitization & Validation:

To ensure system integrity and prevent data tampering, all inputs undergo multiple layers of sanitization and validation before processing.

Security Mechanisms:

- NoSQL Injection Prevention: Use of parameterized queries in MongoDB operations to avoid injection attacks.
- Cross-Site Scripting (XSS) Protection: All input fields are sanitized using Spring's built-in validation and filtering utilities.
- Data Validation: Implemented using annotations like @Valid and @NotBlank in entity classes.
- Global Exception Handling: A centralized error-handling mechanism catches invalid input attempts and logs them for review.

Integration & Deployment Module

Frontend-Backend Integration:

The integration between the React.js frontend and the Spring Boot backend ensures smooth, asynchronous communication for dynamic content delivery.

Integration Process:

1. API Communication: The frontend uses Axios or Fetch API to interact with backend endpoints.
2. Data Exchange: All requests and responses are formatted in JSON.
3. Authorization: Each request to protected routes carries a valid JWT token.
4. Error Handling: HTTP errors (401, 403, 500) are handled gracefully in React with clear user notifications.

Benefits:

- Reduced server load through efficient API calls.
- Enhanced user experience with real-time feedback.
- Seamless synchronization between data layers.

Backend & Database Integration:

The backend communicates directly with MongoDB Atlas, a cloud-hosted NoSQL database platform offering high availability and scalability.

Configuration Highlights:

- Connection established via Spring Data MongoDB using a secure connection string stored in environment variables.
- Indexes are automatically synchronized for improved performance.
- Database credentials and cluster URIs are never hardcoded — managed securely through .env or configuration files.

Data Flow:
React (Frontend) → Spring Boot (APIs) → MongoDB Atlas (Data Storage) → Aggregation Results → React Visualization Components.

Environment Configuration:

The system distinguishes between development, testing, and production environments for efficient deployment and maintenance.

Configurations Include:

- Separate .env files for each environment.
- Dynamic API URLs and database URIs based on environment.
- Logging levels set differently for debugging and production.
- Integration with CI/CD pipelines for automated updates.

Deployment Strategy:

To ensure global accessibility and scalability, the application is deployed across reliable cloud platforms.

Frontend Deployment:

- Hosted Netlify, offering automatic builds from Git repositories.
- Environment variables and build settings are securely managed within the platform.

Backend Deployment:

- Deployed on Render , supporting Spring Boot's JAR execution.
- Includes automatic restarts, logs monitoring, and custom domain setup.

Database Hosting:

- Managed on MongoDB Atlas, ensuring auto-scaling, backups, and performance monitoring.

Deployment Pipeline:

1. Developer commits code to GitHub.
2. CI/CD tools trigger automated builds and tests.
3. Successful builds are deployed to respective environments.
4. Post-deployment monitoring tools track uptime and response latency.

This module—Security & Authentication combined with Integration & Deployment—ensures that the Online Survey & Feedback System remains not only secure but also highly available, scalable, and user-centric. Together, they establish a reliable infrastructure capable of handling real-world usage at academic, research, and corporate scales.