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).

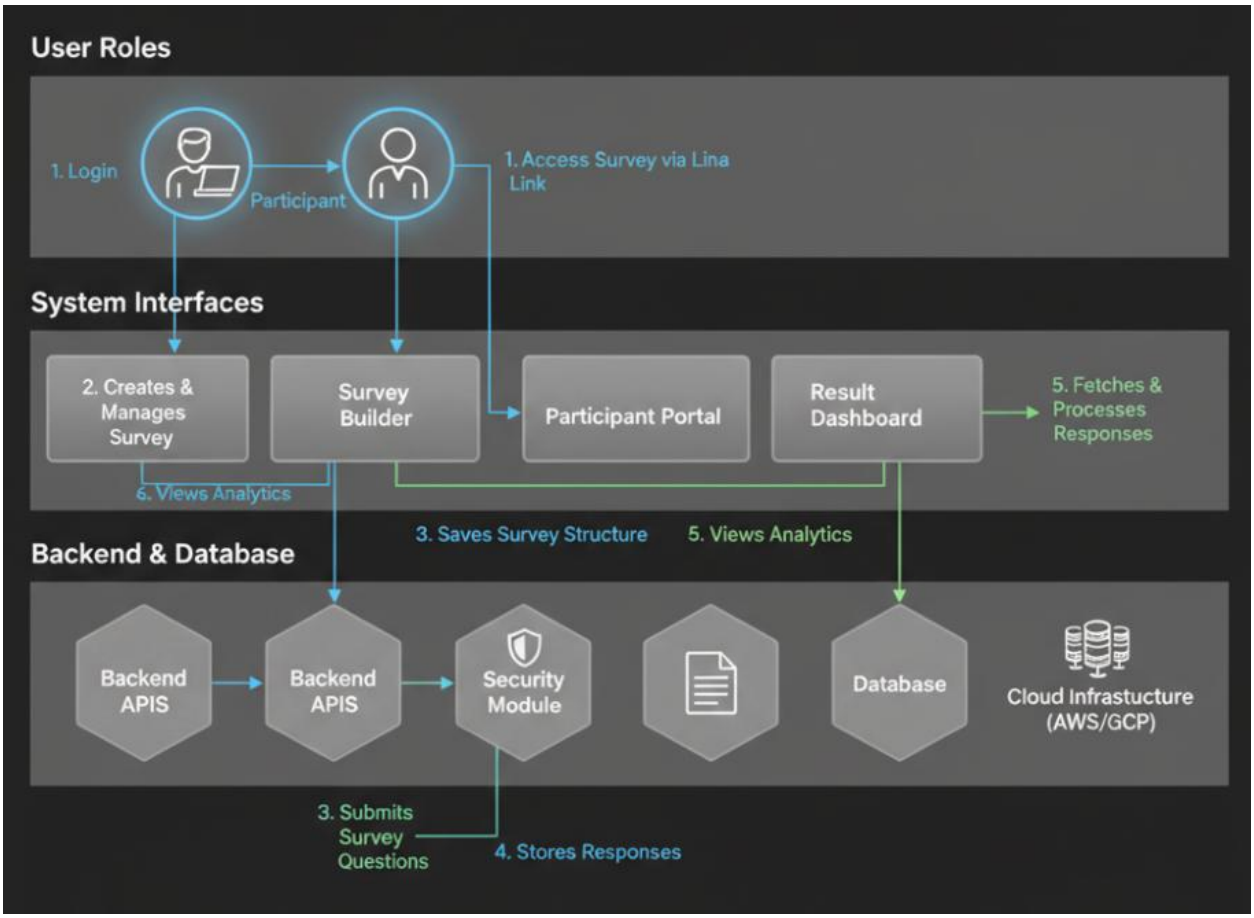
4. Security & Authentication Module

- 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 3: Backend & Database (Spring Boot + MongoDB)

API Development (Spring Boot)

The backend of the Online Survey & Feedback System is developed using Spring Boot, one of the most popular Java frameworks for building enterprise-grade web applications. It provides auto-configuration, embedded servers, and production-ready tools, making it ideal for large-scale, modular applications like this system. The backend acts as the heart of the application, managing all core logic, data flow, user management, and integration with the frontend interface.

The system follows a RESTful architecture, ensuring clear separation between frontend and backend layers. Each API endpoint is designed to perform specific operations and return JSON responses, allowing smooth data exchange with the React.js frontend. The application uses Model-View-Controller (MVC) principles to maintain code modularity and scalability.

Key API Modules and Functionalities:

1. Survey Management APIs (CRUD Operations):
    - Create Survey: Administrators and creators can design surveys dynamically by specifying different question types such as multiple-choice, text-based, and rating scale questions. Data validation and error handling ensure no duplicate or malformed questions are stored.
    - Read Survey: Fetches all surveys or specific survey data from MongoDB based on filters like creator ID, creation date, or survey status. Pagination and sorting are applied for efficient rendering in the frontend.
    - Update Survey: Allows authorized users to modify survey details, add or delete questions, and update deadlines without affecting existing response data.

- Delete Survey: Ensures secure and permanent removal of a survey and its associated responses using cascading delete mechanisms.

2. Response Management APIs:
    - Handles the recording and retrieval of participant responses while maintaining strong referential integrity.
    - Each submission is validated to ensure no duplicate entries are made for the same user and survey.
    - Aggregated response data is processed for analytics and visualization modules in real-time.
3. Authentication and Authorization APIs:
    - User Authentication: Handled using Spring Security, which enforces strict login and signup protocols.
    - JWT (JSON Web Token): Upon successful login, the server issues a JWT token that contains encoded user information and role permissions. All further API calls must include this token in the header for authentication.
    - Role-Based Access Control (RBAC): Divides user privileges into three roles: Administrator (manages system operations), Creator (designs surveys), and Participant (submits responses). Unauthorized access is blocked at both controller and service levels.
4. Security Enhancements:
    - CORS (Cross-Origin Resource Sharing) configuration restricts requests to trusted frontend origins.
    - Data validation prevents NoSQL injection and cross-site scripting (XSS) attacks.
    - Centralized exception handling ensures that errors are logged, categorized, and sent back to the frontend in a user-friendly format.

Database (MongoDB)

The system's database is powered by MongoDB, a flexible and high-performance NoSQL document-oriented database. Its ability to store data in JSON-like documents allows for easy integration with Spring Boot and scalable handling of diverse survey structures.

Database Design and Collections:

1. User Collection:
    - Fields: userId, username, email, password (hashed using BCrypt), role, account creation date, and status.
    - Manages authentication data and permissions for all users in the system.
    - Implements indexes on email and userId to ensure unique identification and faster lookup times.
2. Survey Collection:
    - Stores metadata such as survey title, description, creator ID, creation time, and expiry date.
    - Each survey document contains an array of questions with parameters like question text, type, and options.
    - Dynamic schema allows for varying question structures, supporting both simple and complex surveys.
3. Response Collection:
    - Stores participant submissions linked by surveyId and participantId.
    - Each response is timestamped and validated to ensure the participant has not submitted multiple entries.
    - Aggregation pipelines in MongoDB help compute statistics like average ratings, response counts, and completion rates.

Database Optimization and Performance Enhancements:

- Indexing of frequently queried fields such as surveyId, creatorId, and participantId accelerates query performance.
- Sharding and replication options can be configured for scalability and fault tolerance in large deployments.
- Aggregation pipelines are optimized to provide real-time analytics for administrators.
- The use of Spring Data MongoDB ensures seamless data mapping and repository management.

Integration and Deployment

The backend integrates tightly with the React.js frontend through a well-defined API layer, ensuring efficient two-way communication.

Integration Workflow:

1. Frontend Communication:
    - React components use Axios or Fetch API to call backend endpoints for survey creation, display, response submission, and analytics visualization.
    - The responses are received in JSON format and rendered dynamically on the UI.
2. Authentication Flow:
    - On login, the backend returns a JWT token stored in localStorage or sessionStorage.
    - The token is automatically attached to every subsequent API request header, allowing secure access to protected resources.
    - Token expiration is handled gracefully with refresh mechanisms to maintain user sessions.
3. Cloud Deployment:

- Backend Deployment: Hosted on Render, which provides continuous integration (CI/CD), scalability, and automated restart mechanisms in case of failures.
- Database Deployment: Hosted on MongoDB Atlas, offering features like automatic backups, monitoring dashboards, IP whitelisting, and global cluster access.
- Secure environment variables manage API keys, database URIs, and JWT secrets.

4. Testing and Monitoring:
    - RESTful APIs are tested using Postman and JUnit to ensure reliability and performance under load.
    - Logging is managed using SLF4J and Logback, helping administrators trace issues effectively.
    - Performance monitoring and uptime tracking are achieved via integrated dashboards on Render and Atlas.

This robust combination of Spring Boot and MongoDB ensures that the Online Survey & Feedback System operates with high reliability, flexibility, and security. The architecture is built to scale effortlessly as the number of users and surveys grows, providing real-time insights and seamless integration with the frontend. The backend design not only supports the current needs of survey management but also leaves room for future enhancements such as AI-based analytics, advanced visualization, and automated report generation.