Spam Detector Project

Overview

The Spam Detector project is a Spring Boot application designed to manage user authentication, contact management, and spam reporting. It uses JWT for authentication and provides RESTful APIs for user and contact management.

The key features of the Spam Detector project are:

1. User Authentication:
   - User registration and login using JWT for secure authentication.
   - Logout functionality to invalidate user sessions.

2. Contact Management:
   - Add, update, delete, and retrieve contacts associated with users.
   - Ensure contacts are managed securely and are associated with the correct user.

3. Spam Reporting:
   - Report spam phone numbers.
   - Validate reported phone numbers to ensure they are in the correct format.

4. RESTful APIs:
   - Provide endpoints for user authentication, contact management, and spam reporting.
   - Use Spring Boot to create and manage these APIs.

5. Security:
   - Use Spring Security for authentication and authorization.
   - Generate and validate JWT tokens for secure communication.

6. Data Transfer Objects (DTOs):
   - Use DTOs to transfer data between different layers of the application.
   - Ensure data validation using Jakarta Validation.

7. Service Layer:
   - Implement service interfaces for business logic.
   - Use service implementations to handle user authentication, contact management, and spam reporting.

8. Repository Layer:
   - Use Spring Data JPA repositories for database operations.
   - Perform CRUD operations on user and contact entities.

9. Project Structure:
  - Organize the project into packages for DTOs, entities, repositories, services, controllers, and security-related classes.

10. Technologies Used:
  - Java, Spring Boot, Spring Security, Maven, Lombok, Jakarta Validation.

Technologies Used

- Java
- Spring Boot
- Spring Security
- Maven
- Lombok
- Jakarta Validation
-MySQL
-Hibernate
-JWT

Project Structure

The project is structured as follows:
- `com.spamdetector.dto`: Contains Data Transfer Objects (DTOs) for transferring data between layers.
- `com.spamdetector.entity`: Contains entity classes representing database tables.
- `com.spamdetector.repository`: Contains repository interfaces for database operations.
- `com.spamdetector.security`: Contains security-related classes, such as JWT utilities.
- `com.spamdetector.service`: Contains service interfaces and their implementations.
- `com.spamdetector.controller`: Contains REST controllers for handling HTTP requests.

Key Components

DTOs
- `UserProfileDTO`: Represents user profile data.
- `UserRegistrationDTO`: Represents user registration data.
- `ContactDTO`: Represents contact data.
- `SpamReportDTO`: Represents spam report data.

Entities

1. **users**: Represents a user in the system.

```
mysql> describe users;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | bigint       | NO   | PRI | NULL    | auto_increment |
| email        | varchar(255) | YES  | UNI | NULL    |                |
| name         | varchar(100) | NO   |     | NULL    |                |
| password     | varchar(255) | NO   |     | NULL    |                |
| phone_number | varchar(255) | NO   | UNI | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
```

2. **contacts**: Represents a contact associated with a user.

```
mysql> describe contacts;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | bigint       | NO   | PRI | NULL    | auto_increment |
| name         | varchar(255) | NO   |     | NULL    |                |
| phone_number | varchar(255) | NO   |     | NULL    |                |
| user_id      | bigint       | NO   | MUL | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
```

3. **spam_reports**

```
mysql> describe spam_reports;
+----------------------+--------------+------+-----+---------+----------------+
| Field                | Type         | Null | Key | Default | Extra          |
+----------------------+--------------+------+-----+---------+----------------+
| id                   | bigint       | NO   | PRI | NULL    | auto_increment |
| reported_phone_number | varchar(255) | NO   |     | NULL    |                |
| reported_by_user_id  | bigint       | NO   | MUL | NULL    |                |
+----------------------+--------------+------+-----+---------+----------------+
```

Repositories
- `UserRepository`: Provides CRUD operations for `User` entities.
- `ContactRepository`: Provides CRUD operations for `Contact` entities.

Services
- `AuthenticationService`: Interface for authentication-related operations.
- `UserService`: Interface for user-related operations.
- `ContactService`: Interface for contact-related operations.

Service Implementations
- `AuthenticationServiceImpl`: Implements `AuthenticationService` for user authentication.

- `UserServiceImpl`: Implements `UserService` for user management.
- `ContactServiceImpl`: Implements `ContactService` for contact management.

Security
- `JwtTokenUtil`: Utility class for generating and validating JWT tokens.

Example Usage

User Registration
To register a new user, send a POST request to `/api/auth/register` with a `UserRegistrationDTO` payload.

User Login
To log in, send a POST request to `/api/auth/login` with the user's phone number and password. The response will include a JWT token.

Adding a Contact
To add a contact, send a POST request to `/api/contacts` with a `ContactDTO` payload and the user's ID.

Reporting Spam
To report spam, send a POST request to `/api/spam/report` with a `SpamReportDTO` payload.

Running the Application
1. Clone the repository.
2. Navigate to the project directory.
3. Run `mvn clean install` to build the project.
4. Run `mvn spring-boot:run` to start the application.

API Endpoints

Authentication
- `POST /api/auth/register`: Register a new user.
- `POST /api/auth/login`: Log in a user.
- `POST /api/auth/logout`: Log out a user.

User Management
- `GET /api/users/{userId}`: Get user profile by ID.
- `PUT /api/users/{userId}`: Update user profile by ID.
- `GET /api/users`: Get all users.
- `DELETE /api/users`: Delete all users.

Contact Management
- `POST /api/contacts`: Add a new contact.
- `GET /api/contacts/{userId}`: Get contacts for a user.
- `PUT /api/contacts/{userId}/{contactId}`: Update a contact.
- `DELETE /api/contacts/{userId}/{contactId}`: Delete a contact.
- `DELETE /api/contacts`: Delete all contacts.

Spam Reporting
- `POST /api/spam/report`: Report a spam phone number.

Conclusion
This documentation provides an overview of the Spam Detector project, including its structure, key components, and example usage. For more detailed information, refer to the source code and comments within the project.