## Project Documentation: Online Pet Adoption Platform

### Project Overview

The **Online Pet Adoption Platform** is a web-based application designed to connect animal shelters with potential pet adopters. The platform allows shelters to list pets available for adoption, manage applications, and communicate with adopters. Adopters can browse through available pets, apply for adoption, and track the status of their applications. The platform has three user roles: Admin, Shelter, and Adopter, each with its own dashboard to manage and access functionalities.

### Key Features

- **Admin**: Manages user roles, pet listings, and system settings. Provides analytics on platform activity.
- **Shelter**: Lists available pets, manages adoption applications, and communicates with adopters.
- **Adopter**: Browses available pets, applies for adoption, tracks the application status, and manages personal profile.

### Technologies Used

- **Backend**: Spring Boot (Java)
- **Database**: MySQL or PostgreSQL (for storing pet listings, user data, applications)
- **Frontend**: React
- **Authentication**: JWT for user authentication
- **Validation**: Bean validation using @Valid for input validation

### Setup Instructions

#### Prerequisites:

1. **Java 17** or later installed on your local machine.
2. **Maven** for managing dependencies.
3. **MySQL/PostgreSQL Database** (ensure it is running and accessible).
4. **IDE**: IntelliJ IDEA, Eclipse, or any Java IDE.

#### Steps to Set Up:

1. **Clone the repository**:

bash

Copy code

```
git clone https://github.com/yourusername/pet-adoption-platform.git
```

2. **Configure Database**:
    - Create a database in MySQL/PostgreSQL (e.g., pet_adoption_db).
    - Configure your database connection settings in src/main/resources/application.properties:

properties

Copy code

```
spring.datasource.url=jdbc:mysql://localhost:3306/pet_adoption_db
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update
```

3. **Install Dependencies**: Use Maven to install the project dependencies.

Copy code

```
mvn clean install
```

4. **Run the Application**: Start the Spring Boot application:

arduino

Copy code

```
mvn spring-boot:run
```

5. **Access the Application**: Open your browser and go to http://localhost:8080 to access the platform.

**API Documentation**

**Admin APIs**

- **POST /api/admin/users** - Create a new user (Admin/Shelter/Adopter)
- **GET /api/admin/pet-listings** - Get all pet listings
- **PUT /api/admin/pet-listings/{id}/approve** - Approve a pet listing
- **DELETE /api/admin/pet-listings/{id}** - Delete a pet listing
- **GET /api/admin/analytics** - Get platform analytics

**Shelter APIs**

- **POST /api/shelter/pets** - List a pet for adoption
- **GET /api/shelter/pets** - Get all pet listings for the shelter
- **GET /api/shelter/applications** - Get all adoption applications for the shelter
- **POST /api/shelter/communication/{applicationId}** - Communicate with an adopter

### Adopter APIs

- **GET /api/adopter/pets** - Get all available pets for adoption
- **POST /api/adopter/apply/{petId}** - Apply for adoption of a pet
- **GET /api/adopter/application/{applicationId}** - Track adoption application status
- **PUT /api/adopter/profile** - Update adopter's profile

### Unit Testing

Unit tests have been implemented for the service layer using JUnit and Mockito. The test cases validate the functionality of core services such as pet listing management, adoption application processing, and user role management.

- To run unit tests:

bash

Copy code

```
mvn test
```

### Architecture and Design

The application follows the **Model-View-Controller (MVC)** architecture:

- **Controller**: Handles HTTP requests and responses, uses service layer to perform operations.
- **Service**: Contains business logic, processes pet listings, adoption applications, and user interactions.
- **Repository**: Interacts with the database to perform CRUD operations on pets, applications, and user data.
- **Model**: Represents the data structure for pets, users, and adoption applications.

### Future Improvements

- **User Authentication**: Integrate JWT-based authentication to manage user sessions securely.
- **Search Functionality**: Allow adopters to filter pets by breed, age, location, etc.
- **Notifications**: Implement real-time notifications to inform adopters about application status updates.
- **Payment Integration**: Integrate a payment gateway for adoption fee payments.
- **Admin Dashboard Analytics**: Add more detailed insights and analytics for platform usage and adoption success.

---